CREATING AN AGENT BASED FRAMEWORK TO MAXIMIZE INFORMATION UTILITY

THESIS

John M. Pecarina, Captain, USAF

AFIT/GCS/ENG/08-19

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government.

AFIT/GCS/ENG/08-19

CREATING AN AGENT BASED FRAMEWORK TO MAXIMIZE INFORMATION UTILITY

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science (Computer Science)

John M. Pecarina, BS

Captain, USAF

March 2008

AFIT/GCS/ENG/08-19

CREATING AN AGENT BASED FRAMEWORK TO MAXIMIZE INFORMATION UTILITY

John M. Pecarina, BS

Captain, USAF

Approved:

/signed/	
Dr. Kenneth M. Hopkinson (Chairman)	Date
/signed/	
Dr. Gilbert L. Peterson (Member)	Date
/signed/	
Capt Ryan W. Thomas, PhD (Member)	Date

Abstract

With increased reliance on communications to conduct military operations, information centric network management becomes vital. A Defense department study of information management for net-centric operations lists the need for tools for information triage (based on relevance, priority, and quality) to counter information overload, semi-automated mechanisms for assessment of quality and relevance of information, and advances to enhance cognition and information understanding in the context of missions [30]. Maximizing information utility to match mission objectives is a complex problem that requires a comprehensive solution in information classification, in scheduling, in resource allocation, and in QoS support. Of these research areas, the resource allocation mechanism provides a framework to build the entire solution. Through an agent based mindset, the lessons of robot control architecture are applied to the network domain. The task of managing information flows is achieved with a hybrid reactive architecture. By demonstration, the reactive agent responds to the observed state of the network through the Unified Behavior Framework (UBF). As information flows relay through the network, agents in the network nodes limit resource contention to improve average utility and create a network with smarter bandwidth utilization. While this is an important result for information maximization, the agent based framework may have broader applications for managing communication networks.

Acknowledgments

First, I sincerely thank my wife for granting me the freedom to complete this work. Let these algorithms be a love song to thee. I extend my sincere appreciation to my faculty advisor, Dr. Kenneth Hopkinson, for his guidance and support throughout the course of this thesis effort. Also, I would like to acknowledge the vital contribution of Dr. Gilbert Peterson for his instruction concerning robotics and artificial intelligence.

Finally, but foremost, I give praise to God for his provision, through Him all things are possible.

John M. Pecarina

Table of Contents

Page

List of Figures

Page

List of Tables

Page

List of Abbreviations

AFB	Air Force Base
AOC	Air Operations Center
ATO	Air Tasking Order
DOD	Department of Defense
GIG	Global Information Grid
HAMR	Hybrid Architecture for Multiple Robots
IP	Internet Protocol
JFACC	Joint Forces Air Component Commander
JFCCC	Joint Forces Cyber Component Commander
JNO	Joint Net-Centric Operations
LSP	Label Switched Path
MAS	Multi Agent System
MPLS	Multiprotocol Label Switching
NIM	Network Information Maximization
NS2	Network Simulator 2
NTO	Network Tasking Order
OSD	Office of the Secretary of Defense
QoS	Quality of Service
RSVP	Resource Reservation Protocol
SECAF	Secretary of the Air Force
UAV	Unmanned Aerial Vehicle
UBF	Unified Behavior Framework
TCP	Transmission Control Protocol
TLA	Three Layer Architecture
US	United States
UDP	User Datagram Protocol
USSTRATCOM	United States Strategic Command

CREATING AN AGENT BASED FRAMEWORK

TO MAXIMIZE INFORMATION UTILITY

I. Introduction

Thesis Statement: The quality of information as it relates to the mission can be effectively drawn out of a network using automated, decentralized agent based techniques.

Background

In this era of the information age, there is no doubt that the United States Air Force has become increasingly dependent on the cyberspace domain. As the Secretary of the Air Force (SECAF) stated recently, "A great deal of our combat capability operates in cyberspace: command and control systems as well as the intelligence, surveillance, and reconnaissance platforms that ensure battlefield awareness" [38]. The operations in cyberspace have become so significant that it has become internalized in the Air Force's doctrine. The mission of the United States Air Force is to deliver sovereign options for the defense of the United States of America and its global interests – to fly and fight in Air, Space, and ***Cyberspace***. [34]

With this ever increasing role comes greater challenges. Foremost of these challenges is in the management of the enormous quantities of bits and bytes that represent huge stockpiles of information. This is the most critical aspect of cyberspace, because the availability of information is the very core of the decision making process at any level of command or specialty. The information subspace within cyberspace can be

acted on offensively, using network attack mechanisms, or defensively, as in network defense. However, both operations hinge on the basis that information can be managed in an efficient and effective manner in the first place. Network attack and defense may reduce or bolster this efficiency, but the core competency of information superiority is ensuring an optimal information flow.

Ensuring an optimal information flow is not a trivial ability. It is not simply fulfilled by connecting sensors, users and other data sources with commanders in robust and reliable network architectures and having them communicate, even though this is a daunting task. But merely establishing this would not solve an ever increasing problem. Information has been, is, and will be so ubiquitous that drawing relevant data from a reservoir of data is increasingly complex, time consuming and debilitating. It is a task that the Joint Net-Centric Operations (JNO) address in their Net-Centric Operations Campaign Plan, summarized thusly. The Joint Force and mission partners must have rapid access to relevant, accurate, and timely information, and also the ability to create and share the knowledge required to make superior decisions in an assured environment amid unprecedented quantities of operational data [20].

Much of the work in this problem domain is covered procedurally. Organizations are built on the concept that higher level decisions have greater impact than lower levels decisions. The functional hierarchy and informational hierarchy of organizations tend to grow upwards to satisfy this concept in parallel. At the tactical level, low level actors receive local and specific information. With steps up the hierarchy, and through the operational level, information is globalized and generalized. At the highest, strategic

level, information has been sorted, filtered and fused through the mechanisms of the organization, like a bureaucracy or a command chain. However, the proliferation of information technology has given our adversaries and threats great agility, leaving the traditional construct of organizations cumbersome.

Many military commands have gone to much effort to 'flatten' organizations through various means to bring actionable information closer to commanders. Besides the all too familiar re-organizations that seem to occur every 2-3 years, there have been technological efforts as well. From this author's personal observation, General Cartwright, when he was in his former office as the USSTRATCOM Commander, introduced SkiWeb (pronounced Sky-Web) which he used as a blog site. His charge for the site was that his entire staff, down to the airman, private, and seaman, could answer a blog from the four star general with the intention that information could be quickly escalated to decision makers who needed it. The use of technology to flatten organizations is present on a larger scale in the Department of Defense (DOD), which is in the midst of transforming its vast collection of information-technology systems into an interconnected Global Information Grid (GIG). The GIG will ultimately connect sensors to weapons systems, enable personnel to share information at will, and provide unprecedented levels of situational awareness to commanders at all levels. As Maj Bass indicates in his research, however, if we do not implement the GIG with a proper level of restriction on the flow of information, war fighters risk being overwhelmed not only by too much information but also by information presented at the wrong time, at the wrong level of detail, and without proper analysis and interpretation [1].

With the abandonment of traditional organizational structures to sort, fuse and filter information, technology must fill the gap. The Office of the Secretary of Defense (OSD) recently commissioned an intensive study of information management for net-centric operations [30]. Some of the necessities for technological advancement in the field of information management are:

1) Tools for information triage (based on relevance, priority, and quality) to counter information overload … will become increasingly important.
2) Operators require tools for semi-automated assessment of quality and relevance of information.
3) Advances are needed to enhance cognition and information understanding in the context of missions.
4) A key future capability will be to learn users' context, information needs, and preferences through observation.

Problem Statement

The necessities described in the previous section are the basic motivations for this thesis. However, in order to truly define the problem for the specific context of this work, several scenarios are presented to capture the intended application and focused requirements for this research.

Scenario 1 – In an Air Operations Center (AOC) the commander has several information sources coming to the AOC's central router, each through its own link with a fixed amount of bandwidth. Low bandwidth demand traffic includes sensor inputs that give telemetry and geolocation for aircraft and satellite, and other data link feeds provide

tracking data on friendly and enemy forces. Medium bandwidth demand traffic is comprised of intermittent messages and file transfers from human users outside of the AOC. High bandwidth demand traffic involves multimedia feeds from Unmanned Aerial Vehicles (UAV), camera sensors and very large file transfers. In an ideal setting, the commander would ask to have all this data brought into the AOC and let human eyes decide what is important to the warfighter. However, during periods of peak information flows, available bandwidth capacity is limited, resulting in information loss. The network nodes have limited information to make decisions on what information is allowed to pass through, so decisions become arbitrary. Arbitrary decisions mean that the routing agents make no 'thoughtful' decision, dropping packets regardless of their importance to the mission in a network wide context. This scenario leads to the first problem a system combating information overload would need to solve.

Requirement 1 – Create a system that makes 'thoughtful' decisions on what traffic to drop and what traffic to keep in relation to mission importance.

Scenario 2 – There is a trade off between important information and bandwidth utilization, but often higher utility items must take precedence over lower utility ones. For example, a critical video stream showing a target of interest may occupy a large part of the bandwidth resources of a link. Suppose several other information flows combine to give near optimal utilization of the same link, yet because the video stream is of higher utility, it forces poor utilization. If the system was only concerned with utilization, the second set of information flows would be chosen. There is an obvious tradeoff in this

scenario between utilization and utility, and the system must be capable of finding an appropriate balance.

Requirement 2 – Create a system that balances network resources and mission objectives.

Scenario 3 – The AOC commander has an information network as described above. In this scenario, however, there are several interconnected links between source nodes in the battlespace domain, routing information towards a destination, or sink, node. The source nodes have diverse traffic flows, ranging from low value to high value information and low to high bandwidth demands. Amongst all the nodes in the network, any single source node must compete for the common bandwidth resources of the entire network. If nodes can not effectively communicate in a distributed environment, nodes may reserve all bandwidth at the network's bottlenecks, without considering the utility of competing information flows. Excessive communication may select the highest utility flow, but will also create a less adaptive network with low throughput. The system must have a mechanism to make distributed decisions about its information flows.

Requirement 3 – Create a system that makes distributed decisions about what information needs to flow from source to sink.

Scenario 4 – The battlespace network described above consists of intermittent connections as well as robust connections. One example of these connections involves the scenario where satellites and airplanes pass overhead for a limited but predictable

amount of time. The usage of these connections could be planned for ahead of time using a network tasking plan. Another example of intermittent connections is wireless links that are affected by changes due to mobility or weather effects. Without regard for this reality of this type of hybrid communication model, the network may perform unpredictably and inefficiently. If the goal, furthermore, is to promote relevant data in reaching the sink node, the system must adapt to conditions where bandwidth is in high flux.

Requirement 4 – The system must adapt to periods of bandwidth flux.

Scenario 5 – Building on previous scenarios, this scenario describes the addition of multiple users. In the previous scenario, the AOC Commander is the only user, now there are many actors in the network. Examples of these users are those with tactical level responsibilities amongst the source nodes and users outside of the network, such as an external intelligence analysis site. The additional entities may have distinct and competing mission objectives.

Requirement 5 – Create a system that negotiates multiple, competing mission objectives.

Beyond these 5 issues, there are several other considerations that should be considered important in this field, yet are not specifically addressed.

Scalability ***Security*** ***Fault Tolerance*** ***Reliability***

Availability ***Authenticity*** ***Timeliness***

Vision

A solution that can satisfy the requirements listed above will prove to be an invaluable tool for a commander trying to manage the battlespace. It would give the commander a tool to form the information flow in his environment. True to the SECAF's guidance, such a tool would empower a warfighter in a subset of cyberspace. One could envision a cyber battle analogous to the air battle.

In the current warfighting mindset in an AOC, the commanding authority is the Joint Forces Air Component Commander (JFACC). His major daily efforts are focused around decision making for the Air Tasking Order (ATO) process. The ATO is used to task and prepare air units to accomplish the overall mission. While analyzing how the previous ATO was played out, the JFACC watches as the current ATO is managed, makes necessary changes to the next day's ATO and creates an ATO for three days out. One of his major tools in affecting the development process is in assigning weights of effort to the air battle's many objectives. For instance, if the supreme commander feels that protecting a major urban center from attack is a priority, then the air component commander guides the strategy to task by assigning a certain portion of his force to it. This guidance helps war planners devise air tasking options needed to reach the objective.

It is feasible to envision a movement towards a Network Tasking Order (NTO), similar to an ATO, which would guide communication assets in a theater of operations. To provide an illustration about how this may occur, a Joint Forces Cyber Component Commander (JFCCC) could assign weights of effort to different mission goals in cyberspace. For example, if protecting an urban center is a priority, the JFCCC will see it

as a necessity to ensure information superiority over that area to support the other commanders. Thus, he would scale his weight of effort to give planners a queue to favor missions that support this objective. Mission planners could subsequently coordinate to provide more airborne or land based communication assets to provide extra bandwidth in that area of the battlespace.

However, due to the incredible dynamics associated with network communications, it may be infeasible to expect a daily tasking order to be carried out in any manner close to what is planned. As bandwidth and connectivity vary over time, battlespace communication nodes will require protocols that can negotiate amongst themselves to split additional bandwidth or, when bandwidth is lower than expected, remove low-priority communication tasks. In a real sense, the NTO will be a dynamic plan that may change on the order of seconds, taking into account changes in satellite orbits, airborne and seaborne platforms, as well as land based lines that may suddenly appear or disappear in a battlespace environment. Given that an NTO is a likely step forward, real-time mechanisms must be in place to detect deviations from preplanned network bandwidth and link availability. The network must self-adapt, bearing in mind the commander's intent, his weights of effort. It is the goal of this research to devise a system which could provide the interface for such a commander to have this input, and expect it to be reasonably carried out.

There are many research areas that require solutions that match the requirements and vision that will bring a cyberspace commander more control of his network. A solution needed within information maximization involves creating a multi-agent system

to control the network. With the backdrop of the motivation for this research, this thesis creates an agent based framework to maximize information utility.

Preview

In summary, the military is moving towards a more information-rich environment for the warfighter. While this has the potential to revolutionize the way that wars are fought, getting the right information to the right decision makers becomes extremely difficult as attempts are made to scale up to an entire battlefront. UAVs, sensor networks, advanced satellite data, and publish-subscribe systems all have the potential to be a great aid, but they require bandwidth management and topology control on a much wider and more complex scale than exists at the present time. This thesis will help to move closer to a world where warfighters can have the information they need, in the form it is needed in, and at the time they need it. The work will simultaneously help to move to a point where the information that is not needed, that wastes time, bandwidth, and energy is blocked to allow the critical data to get through.

This chapter provided motivation for the research area and presented a brief description of the problem. Chapter II will present the subject matter in more depth to show what research that has already been conducted in this area. It will also describe how this research is different from previous research on the same topic. Chapter III gives a full explanation of the methodology and details the approach used in conducting the experiments. Chapter IV compares the different experiments conducted and presents the results is a logical manner. Finally, Chapter V summarizes the experiment results, explains the significance of the research, and presents areas for future research.

II. Literature Review

Network information maximization, simply stated, is the ability of a network of queues, schedulers and routers to produce the most useful data to a user or users given time and bandwidth constraints. The most useful data for any window of time consists of the optimal set of information flows in terms of information utility. Information flows are not merely file or message transfers. They are much broader than that, representing long standing, heterogeneous, and sometimes discontinuous blocks of data answering specific information requirements for a user. Information utility is a measure of the importance of a piece of information to a user who desires it. In the literature, information utility can be synonymous with the terms "information value" [27] or "information measure" [15] and "relevance" [26]. It is somewhat synonymous to the term "priority" or "ultimate priority" [22], though priority has been closely associated with timeliness in multimedia. Timeliness requirements may not be the sole decision point for maximizing useful information from a network. This paper chooses to use the term information utility for this association with usefulness. The goal of the network information maximization problem is to maximize the aggregate utility of the set of information flows presented to a user, which can be defined as the sum of all utilities of all information streams. A related metric, average utility, can be defined as the aggregate utility divided by the number of packets received over a time interval. Average utility is also maximized as aggregate utility is maximized.

Network information maximization (NIM) is a problem within network management that is concerned with the research areas of information classification, scheduling, resource allocation, and providing quality of service (QoS). A comprehensive solution requires a system that accounts for all of these aspects. Although the work presented in this thesis concentrates on resource allocation, it makes assumptions about classification and scheduling of information flows that call for some elucidation. The resource allocation solution implements an agent based approach, built with reactive robot control architecture that works on top of a QoS framework. Keeping a comprehensive view, this chapter illuminates information flow classification, a solution for scheduling, an agent based architecture to exploit information and mechanisms for QoS.

Classification for Information Maximization

The classification of information flows into a meaningful description of its value is one critical competency of managing an information system. As will later be shown, classification can support the ability to index information flows for scheduling and the allocation of resources and can help define a class of service for QoS. It is important to use a quantitative descriptor for an information flow. The sources of that descriptor can be numerous, but this work focuses on mission association, timeliness, and user input to determine value. Finally, these classifiers can be combined to form an information utility for an information flow. Appropriate information classification relies upon choosing a type of descriptor and the sources of classification to combine a usable index for information.

Descriptors

In non-specific terms, information utility may be a qualitative description of the information, where one would expect the adjectives low value, fair value, high value. For example, Huang [19] asserts the terms critical, essential or non-essential class. However, if only one qualitative descriptor is used, it is very difficult to capture a user's true intent in deciding what information needs to be seen by the commander. To illustrate this, consider a commander interested in maintaining air superiority over Baghdad. Assuming air superiority has already been established, the commander may wait on certain information flags to alert him to the possibility of an escalating threat. Some information flags are obvious, like detecting mobile surface to air missile sites, enemy aircraft, and jamming events. Information drawing these conclusions would garner a high value rating, because they explicitly match his intent. Other events in the area over Baghdad are more subtle, such as enemy troop movement, which may or may not have a capability to hinder air operations. Information in this category may have a fair value rating. Finally, many seemingly unrelated events, such as reports of snipers in buildings, may occur. This information would likely have a low value rating. If a commander asked to see all information, information overload may occur. A new threshold could be set to receive only high value information, but the risk of missing important information from a lower perceived value increases. Also, information from lower value events when combined with other low value events could be missed when setting a qualitative threshold. For instance, snipers in several building in strategic topographic locations could be setting themselves up for a coordinated jamming attack while enemy troops position themselves

for an urban ground battle. If their coordination succeeds, their combined efforts may overwhelm ground forces when aircraft are unable to give close air support and deliver precision guided munitions.

The example illustrates a problem that qualitative descriptors have, arbitrating between quality levels. A quantitative description can help alleviate the problem, using numbers instead of words to describe the value of information. Low, fair, and high can be replaced by 1, 2, and 3. These numbers can be adjusted in the presence of other events. In the example, a sniper being spotted in Baghdad may increase the next spotted sniper to 1.2. Events can be combined with other events and given an average or intermediate value which is more meaningful if they could only be described as low value.

Kwiatkowski [22] proposes a combination of 3 qualitative and quantitative descriptors that are combined into a single quantitative descriptor. The first is a mission identifier that contains a weighting based on relative importance assigned by a commander or a policy in place on the network. The second is a precedence value, given the values Routine, Priority, Immediate or Flash. These values correspond to the time, order of minutes and hours, that the information is needed. They would correspond to time constraints written in military procedure or doctrine. The third descriptor comes from a user-perceived priority. These descriptors are combined into an ultimate priority by a network policy, which may assign a value using the descriptors and factors like resource demand or traffic type. This research parallels Kwiatkowski's notion of descriptors, but describes the attributes in different detail. This thesis asserts that mission association, timeliness, and user input are used to calculate an information utility value.

Mission Association

In a military environment, mission objectives and goals are the most obvious source of value for information utility. Hintz and McVey [15] attempted to maximize the flow of information in a group of sensors by minimizing the uncertainty of the perception of the world held by a mathematical model and the world itself. While this view maximizes information flow, it lacks a representation of information needs. While maintaining the model of reducing uncertainty to maximize information flow, Hintz and McIntyre [14] developed the concept of a goal lattice, to extract sensor task weightings from mission objectives. A software system called GMUGLE decomposes a goal into sub-goals and eventually into tasks. The top level goal is always given a weight of one. Every level down allocates the parent's weight among the different sub-goals. The bottom level consists solely of tasks, weighted to reflect how they support the top mission goal. This weighting is then used to decide where to focus information foraging efforts. A mission oriented goal lattice developed by McIntyre [25] is shown in Table 2-1 and Figure 2-1. Taking seventeen mission goals from Air Force doctrine, he produced a goal lattice to weight sensor tasking. The top goals in the table correspond to the top goals in the goal lattice, continuing as the graph moves to the bottom goals. The bottom three goals of track, identification (ID), and search are reflected in the bottom layer of the goal lattice, and their respective weights are used in choosing a task. Hintz and Malachowski [13] extend this model to dynamically build goal lattices to fit real operations environments better.

Table 2-1: US Air Force doctrinal goals from "In Harm's Way" scenario [25].

Goal Number	Goal	Included Goals
1	to obtain and maintain air superiority	2, 3, 4, 5
2	to minimize losses	6, 7, 8
3	to minimize personnel losses	6, 7, 8
4	to minimize weapons expenditure	6, 8
5	to seize the element of surprise	8
6	to avoid own detection	9, 10
7	to minimize fuel usage	10, 11
8	to minimize the uncertainty about the environment	12, 13
9	to navigate	15, 16
10	to avoid threats	15, 16
11	to route plan	15, 17
12	to maintain currency of the enemy order of battle	14, 16
13	to assess state of the enemy's readiness	14
14	to collect intelligence	15, 16, 17
15	to track all detected targets	
16	to identify targets	
17	to search for enemy targets	

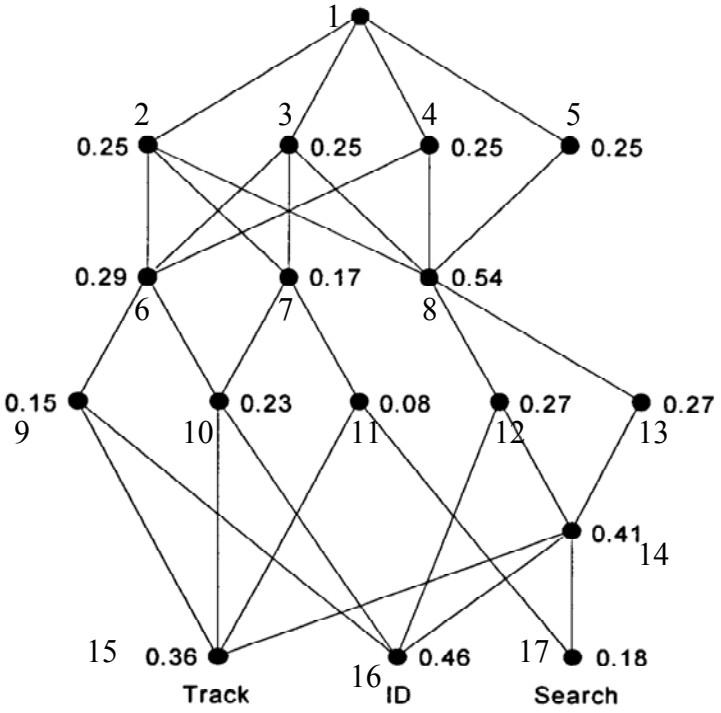

Figure 2-1: Goal Lattice for "In Harm's Way" scenario [25].

In the NIM, a goal lattice method could decompose the commander's goals to sub-goals in tasks. As information flows enter the system, they would be registered with a mission goal or task to receive a weighting reflecting their importance to the mission. If the information flow supports more than one goal to a varying degree, it would be registered with a weighted average for the various goals it supports. An information flow's weighting would give it a quantitative mission association value. The goal lattice construct enables a commander to manage a network using the NTO. Being able to weight goals and tasks, a commander can see how their intent is being disseminated to an army of network devices and can rearticulate mission objectives accordingly.

Timeliness

In some cases, timeliness is the most critical attribute of information. Intrinsic to the concept of an information flow is that the entire stream of information is of value to a user at or before a deadline. Keshav [21] explains that utility can be represented as a function of the time delay that it takes for information to reach a destination. One possible utility function is $u(t) = U - t$. If the time delay were infinitely small, the utility of receiving information in this case would be U. As t increases, u(t) decreases. When $t > U$, the utility of receiving such information would actually begin to be negative, since it has no value and ties up resources in order to be sent late. Information flows that consist of voice, video, or extremely critical information may be classified into timeliness clusters of immediate. Other clusters, like priority and routine, could express increasing levels of delay tolerance.

User Input

User input may provide another means of classifying information. One form of user input is user driven segmentation, described by Berkhin [2], refers to the process that utilizes expert knowledge regarding the importance of certain sub-domains. In this case, user knowledge could be derived from an NTO process in which the commander and his staff create rules that could be incorporated into an expert system. For example, a commander could assign a rule linking the appearance of a known terrorist in the presence of other explicitly named terrorists to a mission critical cluster. User input should be limited, however, because the capability for a user to react to dynamic events in information flow is also limited. Therefore, for simplicity, three categories, Mission Critical, Mission Essential and Mission Non Essential are adopted for user input.

Popularity is another form of user input. Search engines on the World Wide Web provide powerful keyword search mechanisms that may be applied to the battlespace network. Search engines can weight previous page hits to guide future searches. One way to use search engine concepts is to reinforce value in popular information flows, allowing user input to alter information utility to exploit useful information flows. Popularity tends to focus on a commonly accessed subset of all available information. Thus, popularity must be limited so exploration is balanced among information flows [18].

Information classification is a vital piece to network information maximization. Proper classification supports the ability to index information flows for scheduling and the allocation of resources. This work focuses on mission association, timeliness and user input to determine classification. After classification, these three attributes can be

combined to form an information utility for an information flow during the scheduling process.

Scheduling for Information Maximization

The scheduler determines an information flow's utility value to prepare it for resource allocation and routing. The three attributes determined by a classifier are combined to form an information utility, used as an index in scheduling. This section describes the information flow scheduling problem and maps it to the restless bandit problem. The restless bandit problem is formulated, then its complexity is described and an approximate solution is shown. This section concludes with an applied solution to the information maximization domain. The information flow scheduling problem, similar to the restless bandit problem is solved with an approximate greedy solution that supports the concept of indexing information utilities.

Information Flow Scheduling Problem

The information flow scheduling problem solver must choose an optimal set of information flows for a user. If only one user existed, such as an AOC commander, the network goal is to select the set of commodities that produces maximum information utility to the AOC commander. For each node in a network, a scheduling device is responsible for selecting a set of information flows to forward to the next node that will contribute to the network goal. The multi-armed bandit problem further characterizes this scheduling problem.

The multi-armed bandit problem [28] is based on an analogy to a slot machine (the one armed bandit), but with many levers. Each lever is associated with an expected

distribution that comes with each pull of the lever. If the lever is activated, a user receives a reward. Without initial knowledge of the distributions, a gambler must select a subset of levers that maximizes distributions over time. The gambler faces the dilemma of exploiting one set of levers for their value or exploring other sets of levers to gain better distributions. In the information scheduling problem, each network scheduling device plays the role of a gambler. The scheduling device chooses from a set of information flows to maximize utility in the long term.

The multi-armed bandit problem is solvable in polynomial time using a priority index approach. The approach is to assign a priority index to every lever corresponding to its current state. The optimized set at any given time includes all the levers with the largest indices in the current state, selected greedily [3]. However, for this research, a slightly adapted model of this problem must be used, because the multi-armed bandit problem assumes that there is no incurred cost and no change in the state of an item if it is not activated [31]. On the contrary, not choosing an information flow as a service deadline approaches may incur cost in its utility or alter its relative popularity. Therefore, this paper presents an extension of the multi-armed bandit problem, the restless bandit problem.

Restless Bandit Problem

Bertsimas and Nino-Mora [3] presented a formulation of the restless bandit problem. Let there be a collection of N projects. Project $n \in N = \{1, \ldots, N\}$ can be in one of a finite number of states $i_n \in \mathrm{I}_n$, for $n = 1, \ldots, N$. At each instant of discrete time,

t = 0, 1, 2,...,∞ exactly $M < N$ projects must be operated. Let $a_n(t) = \{0 , 1\}$ denote that project n is active at time t. If project n, in state $i_n(t)$, is in operation, then an active reward R^1_i is earned, and the project state changes with an active transition probability. If the project remains idle, then a passive reward R^0_i is received, and the project state changes with a passive transition probability. Rewards are discounted in time by a discount factor $0 < \beta < 1$. Projects are to be selected for operation according to an admissible scheduling policy u: the decision as to which M projects to operate at any time t must be based only on information on the current states of the projects. Let U denote the class of admissible scheduling policies. The goal is to find an admissible scheduling policy, Π*, that maximizes the total expected discounted reward over an infinite time horizon.

$$\Pi^* = \max E_u \left[\sum_{t=0}^{\infty} (R_{i_1(t)}^{a_1(t)} + \ldots + R_{i_n(t)}^{a_n(t)}) \beta^t \right]$$

Later, this section describes how this formulation applies to the NIM scheduling problem through substituting projects for information retrieval.

Computational Complexity

Papadimitriou and Tsitsiklis [31] described the complexity of optimal queuing network control and specifically address the restless bandit problem. They proved that the problem of finding an optimal control policy in a multiclass closed queuing network is an intractable problem. Not merely this, but also that the problem provably requires exponential time for its solution, or EXP-complete. If both routing and the service times

are deterministic, they show the problem to be PSPACE complete. Finally, they show that the restless bandit problem is PSPACE complete even for deterministic problems.

Approximate Solutions

The well-known optimality of Gittins priority index rule was applied to the restless bandit problem by Whittle in 1988. However, in 1990 Weber and Weiss found instances of the restless bandit problem that did not satisfy a certain indexability property. Bertsimas and Nino-Mora produce a second order linear programming relaxation to approximate a heuristic they called the primal-dual heuristic. Using the primal-dual heuristic in place of the Gittins priority index, they found excellent performance in computational experiments that could be improved with stochastic optimization methods, like Genetic Algorithms [3]. Later, Nino-Mora [29] pared down the linear programming solution into an adaptive greedy algorithm that calculated an index for each item state.

Solution in Information Maximization

Huberman [18] built upon this adaptive greedy algorithm and applied it specifically to an information maximization problem. His viewpoint was that the average user is bombarded with information overload while searching the web for useful information. He claims that search engines can often employ arbitrary methods for listing the top search items instead of ranking items by user preference. The average user becomes saturated very quickly, often choosing only the first few hits in a search, though sometimes the best results exist much later in the list of results. In his formulation, he mapped the problem of optimizing the information one gets from exploring web pages or any other digital content to Bertsimas' optimal allocation of effort to a number of

competing projects. His mathematical model varies from Bertsimas in semantics, but maintains the same idea of maximizing reward over time.

At any time t, n items exist, but only m of those items can be displayed at any given time. In every time step, it is assumed that the system can update its list of items to display. Let $A_n(t)$ be the set of all items, i at time t, where $A_n(t) = 1$ if it is displayed at time t, and 0 otherwise. Let r_i be the total expected utility of any item. Let I be the set of all states that any item can be in at any time. Let U be the set of all users, u and let β be the future discount factor such that $0 < \beta < 1$. Then, the objective function to maximize total expected utility for all users is given below.

$$\max_{u \in U} E_u \left[\sum_{t=0}^{\infty} \sum_{m=1}^{n} (r_{i_m(t)} A_m(t)) \beta \right]$$

Huberman simplifies the problem by creating a dual speed assumption. This means that active actions, choosing an item, causes a faster change in state than, passive actions, not selecting an item. This is an important assumption because it gives the problem the property of indexability. It makes sense in this environment because search engine hits that are viewed are far more likely to be selected or not selected again after they are viewed. This is a plausible assumption in the network information maximization problem as well, because once information is viewed, users can better categorize it as something they prefer or no longer prefer to see. Thus, this thesis makes the same assumption as Huberman does.

Huberman continued in solving the information maximization problem by determining a set of constants which he uses in the Bertsimas-Nino-Mora greedy

algorithm to calculate a set of indices for each of the states in *I*. These indices were rank-ordered to produce his top information retrievals. In his experiment, Huberman created a state space that consisted of a two attribute vector, containing a 5 star rating system and a 5 access level rating (based on number of website hits). His state space held 25 states, and using the adaptive greedy algorithm, the optimal set of search items was found. One unexpected result in his experiments showed that items that were rarely seen were in states that held higher value than items that were seen occasionally but had mediocre ratings. The author claimed this is due to the fact that the algorithm gives high index values to potentially valuable states. Such a property would be desirable in information retrieval methods to encourage exploration of all available information, rather than focusing only on what has been seen.

In the NIM problem, the classification process described in the previous section would create a state space that would consist of 100 discretized points describing a percentage towards mission support multiplied by the number of timeliness clusters multiplied by the number of user input clusters. Using the adaptive greedy algorithm presented by Bertsimas, the states could be indexed and sorted on their information utility. After scheduling, resource allocation and QoS routing mechanisms ensure the information flows reach their end destinations.

The study of the information scheduling problem gives an important result. Huberman's formulation shows that information has the property of indexibility. This is critical to the success of the resource allocation mechanism, which relies on an accurate indexing of information flows to ensure the highest aggregate utility for network users.

Resource Allocation for Information Maximization

Classification provides the attributes that are used in scheduling through indexing. However, a problem that still exists comes from a system wide allocation of bandwidth so bottlenecks in the network will appropriately assign bandwidth and time to information flows that are farther upstream or to prevent congestion at bottlenecks downstream. This must be done in a dynamic way, because the utility of information flows can change drastically from one moment to the next, affecting changes in resource allocation. Also, the addition of information flows and loss of bandwidth can change resource allocation. To react dynamically, an agent based approach is introduced. A hybrid agent architecture can perform tasks for resource allocation by coordinating with fellow agents in a multi-agent system.

Without significantly altering a network, the combined utility of information coming from a network can be greatly increased simply by placing priority queues at each network node. By doing so, the lowest utility flows are dropped before higher utility flows, ensuring that the most useful of all flows get to the end user. The problem with this simplistic arrangement is that it does not ensure that the second highest utility flow gets to the end user, since it may have been dropped because of contention with the highest utility flow. A later introduction of a lesser utility flow produces a lower aggregate utility. This scenario is shown in Figure 2-2.

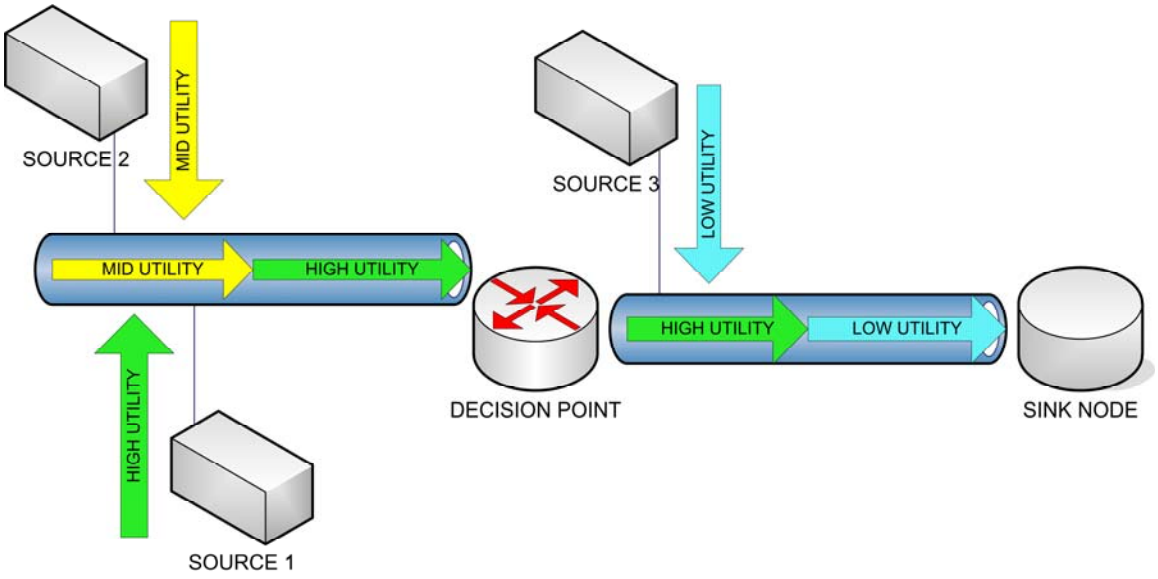

Figure 2-2: Suboptimal aggregate information utility. A high utility flow and a low utility flow are relayed through a network while the decision point arbitrarily drops the mid value utility flow.

Another problem comes when an arbitrary resource allocation process merely looks at utility but not at utilization of network resources. In the scenario depicted in Figure 2-3, source 2's low utility flow vies for bandwidth against the high utility flow and a flow heading to an alternate sink node. Here, the low utility does not gain access to the sink node because of competition with the downstream mid-utility flow. Yet, the low utility flow still utilizes the bandwidth that the alternate source could use to reach the alternate sink. The underutilization of the link to the alternate sink indicates poor utilization of network resources.

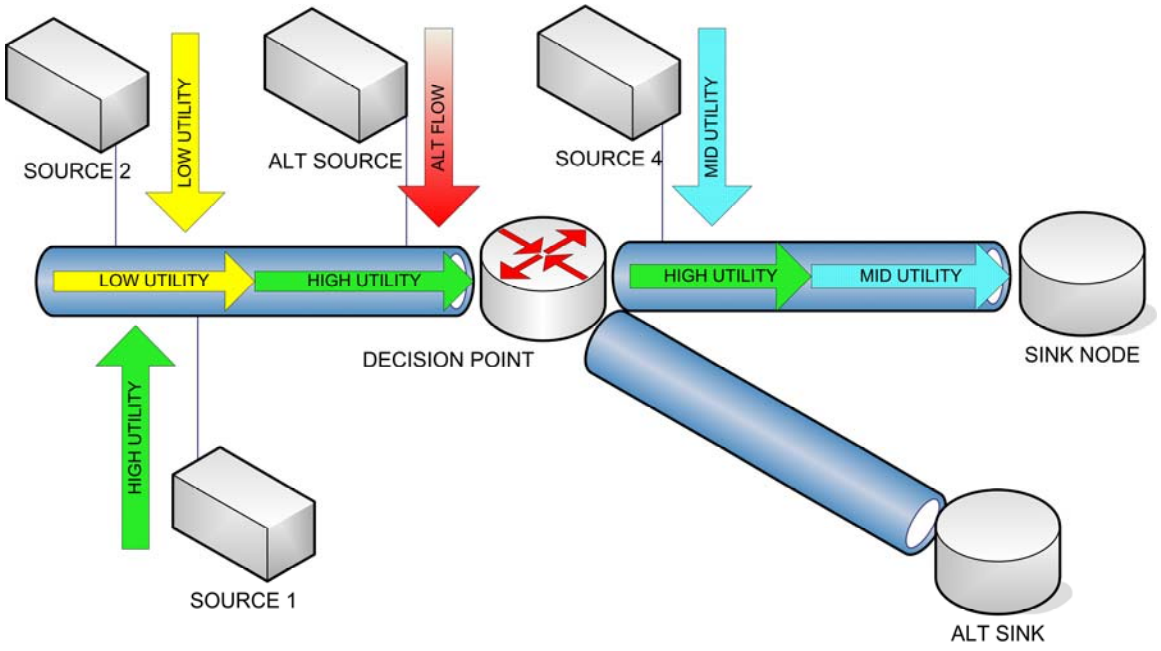

Figure 2-3: Poor utilization of network resources. An alternate flow attempting to reach an alternate sink in the network is unable to compete for the bandwidth of a shared congested link. Since high and mid utility flows occupy the final link to the sink, the low utility flow unnecessarily utilizes the shared congested link, resulting in poor utilization of the link to the alternate sink.

Agent Architectures

An agent is an entity that uses machine logic to analyze state and perception to act on its environment. This paper uses the term robot and agent interchangeably, but the term robot typically describes an agent in the physical or simulated real world. Still, many typical robot control architectures can be used in agents. This work proposes that agents using a three layer architecture (TLA) [11] can perform the duties of resource allocation.

Before 1985, traditional implementations of robot control architectures exhibit the same issues that centralized control of networks face. Typically, a planning module or

"brain" of the robot created an action based on state information through extensive planning algorithms. The robot executed actions through its actuators. If the environment is highly dynamic, planning would be intensive. Traditional implementations suffered from excessive time delay in dynamic environments [11]. In the same way, the dynamic environments present in wireless networks with dynamic topologies hinder the centralized control of a network. Time delay and uncertainty in distributed systems create very difficult or impossible optimization problems. As robot control architectures have modernized, the lessons learned from their evolution can be applied to communications networks.

Reactive Control Architecture

Modern ideas about how robot control architectures work have a basis in the work of psychologist Valentino Braitenberg [5]. Braitenberg described proposes that simple systems can exhibit complex behavior. First, he described how robots can dynamically react to an environmental stimulus by just wiring sensors to motors. As a sensor measures the increasing presence of a stimulus, the robot accelerates toward the stimulus and decelerates as the absence of it increases. The opposite may also occur when the wires are crossed. The important thing to note is that this property of locomotion sets the stage for emotion. For example, a robot that charges a stimulus demonstrates aggression. If the robot flees the stimulus, it demonstrates fear. Next, Braitenberg adds sensors that react to different stimuli to build a multisensorial vehicle that exhibits a sense of knowledge or values. Then, he introduced a motor that was not simply on or off, but instead was powered proportionally based on the intensity of the stimulus. The robot that was created

by this development had very complex trajectories that can be perceived as instincts and decision making, which gives the observer an impression that thought was occurring.

Braitenberg suggests an important point in the field of robotics, that a robot can display characteristics and behaviors that mimic those of real living creatures. The motor/sensor organisms describe an elemental view of robots and illustrate how they exhibit emotion, value, knowledge and logic. He also showed how simple behaviors can combine to form more complex ones.

Rodney Brooks [6] published this idea concurrently as it applied to robot control architectures. Brooks described robot control architecture based on the behaviors the robot must exhibit. He emphasized simplicity, even in a complex environment, commenting that simple implementations are more manageable than complex, more capable ones. Brooks illustrated his ideas about the robot control architecture, based on a behavior model, which built levels of competence. Starting with the lowest level, obstacle avoidance, each level built on the previous one, adding mobility, sensing, and planning until a very sophisticated, reasoning robot emerged. In his design, the highest level of competence overrode the behavior in the levels below it in what he described as subsumption. The delineation in the levels allowed Brooks the ability to engineer solutions in an incremental fashion, adding new levels of competence to the control architecture when they were called for or available.

The subsumption architecture provided four advantages. 1) it allowed multiple goals, as long as a higher level goal did not cancel a lower level one, 2) it allowed multiple sensors to operate, through fusion or centralization, 3) it brought a limited

robustness, where failure at one level meant falling back to a lower one, and 4) it was additive, since each level came with its own processor. To implement each layer, Brooks used a finite state machine implementation, claiming that each layer only used the parts of the traditional planning model that it required, thereby speeding processing time.

Brooks' article was important because it provided a new understanding of robot control that freed it from traditional linear planning methods. The new method created the reactive robot that could act faster on changes in its environment. According to Gat [11], the reactive robots had huge successes early on, running circles around the traditional planners. However, in 1989, the robot Herbert demonstrated what was perceived as the "capability ceiling" in the subsumption style. The fatal flaw of subsumption was an inability to deal with complexity, owing first to not being able to separate the layers into prioritized levels and second to not being able to keep state information. The problem with the pure planners was its slow speed because of its state intensive planning and searching. On the other hand, the pure reactive systems suffered from having no state information which caused it to be unreliable and incompetent in complex environments. However, Gat argued that these layers were complementary and that both have value, yet a third layer was needed to reside between them.

Three Layer Architecture

Gat [11] presented a seminal paper on TLAs in 1998. TLAs consist of control, sequencing and deliberative layers. The basic idea is that a low level control layer handled functions which reactive architectures were good at, such as wandering and obstacle avoidance. At the top, a deliberative layer performed planning functions which

required time. Between these layers, Gat identified a sequencing layer that selected an appropriate reactive behavior based on the perceived state of the environment. This layer made the robot act more intelligently than if it used a reactive approach alone, and faster than if it was only a deliberative planner.

To make the TLA more flexible, Woolley [37] introduced the Unified Behavior Framework. The UBF works at the controller level of the three layer architecture. Several low level behaviors run concurrently at the controller layer, producing instinctive actions to a perception of the environment. Each action also produces a vote that indicates how strongly each behavior module believes its action should be implemented. An arbiter selects an action for a situation based on the behavior votes. Arbitration between voting behaviors can be done with a simple or fusion approach. The simple, or winner take all approach chooses one action based on the highest vote. The fusion approach can take many forms. One form, command fusion creates an action that is a weighted average of all inputs. Another form, utility fusion selects the highest voting action but adds actions to unused resources in vote order. The simple arbitration scheme can be quite adaptive to dynamic environments, but lack the complex, emergent behavior property of the fusion techniques.

Hooper [16] portrayed how the UBF fits into a TLA. He asserts that the Controller is responsible for selecting and activating behaviors through the UBF. The Sequencer selects a set of candidate behaviors for the UBF by analyzing current tasks and state information. A depiction of how the UBF fits into the TLA is shown in Figure 2-4.

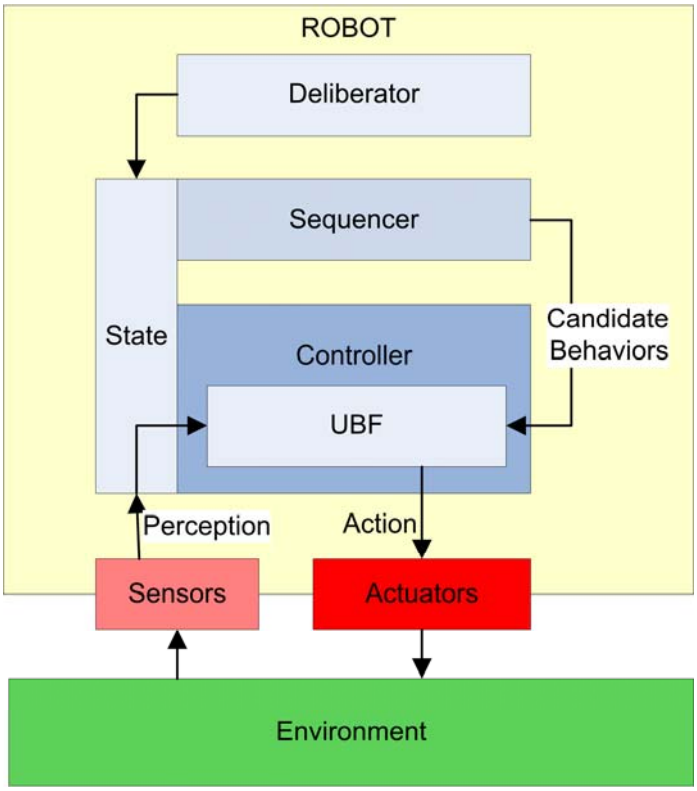

Figure 2-4: Three Layer Architecture including the UBF.

Brooks, Gat, Woolley and Hooper applied their concepts to robots in the physical (or simulated physical) world. Their goals were aimed at creating robots that could navigate around obstacles and perform tasks in an environment that mimicked the physical world. This research attempts to apply robot control concepts to a virtual computer network domain. The deliberator layer allows for input (i.e. a network tasking order) from network owners, (i.e. commanders) to alter a network based on mission objectives. The sequencer layer can take decomposed goals from the deliberator to build a library of behaviors that are sufficiently capable of performing a task. The controller layer ensures task execution. The end result is an agent deployed in every node that

considers the state of the network and selects actions that seek to maximize the aggregate utility of the network flows, while balancing the efficient usage of resources.

Yet, because of the networks constraints of bandwidth and time, agents can not act strictly independently. A single controller can use global knowledge to arbitrate among several agents to compete for bandwidth reservations to cross the network. The obvious difficulty with this method is that it forces a centralized network architecture, which will not scale well and is not fault tolerant. Since decisions are given to local agents for the purpose of decentralization, coordination is necessary to select the optimal set of information flows. Therefore, information maximization requires a multi agent system that can perform resource allocation through cooperative behavior. Cao [7] defines cooperative behavior in this way "Given some task specified by a designer, a multiple-robot system displays cooperative behavior if, due to some underlying mechanism (i.e., the “mechanism of cooperation”), there is an increase in the total utility of the system." The next section describes multi agent systems in the context of network information maximization.

Multi-Agent Systems

A Multi-Agent System (MAS) is a generalization of a multi-robot system, but many of the same characteristics are present in both. Like many multi-robot systems, a MAS has no global system controller, asynchronous computation, decentralized data, and agents without complete information or capabilities to solve a problem [35]. A multi-agent system can also be described as a distributed system. According to Wang [36], a distributed computing system contains many computing devices that are geographically

separated, where each computing device consists of processing and storage facilities as well as a subsystem that supports inter-process communication. Cao calls the multi robot system a special case of a distributed system [7], so the taxonomies of multi-robot systems can be applied to multi-agent and distributed systems. The taxonomies of Cao and Dudek [9] characterize several traits for a multi-robot system which can be applied for the NIM problem.

Cao [7] offers four traits of multi-robot systems. The first trait is that they are either centralized or decentralized. If they are decentralized, then the system is either hierarchical, meaning local centralization, or distributed, meaning all agents are equal in control. The NIM has disregarded centralization as infeasible, so its solution must be decentralized. Further, it is hierarchical since nodes closer to the sink node or at bottlenecks will have greater impact on information flows than others. The greater impact is present because leaf nodes have less access to information while interior nodes may have to decide how to route a larger percentage of information. The second trait describes whether all nodes are homogenous or heterogeneous in regards to capabilities. In the NIM, all nodes have the same capabilities. However, bandwidth capabilities may hinder some nodes from functioning as well as others. Nevertheless, in describing this Multi-Agent System functionality, bandwidth capability is disregarded. The third trait in Cao's taxonomy is the agents' communication structure. Agents interact via the environment, sensing or communication processes. Communication can be either directed or broadcast. The existence of the agents on a communications network implies that communication is probable, and could be directed or broadcast. Wang [36] proposes 3 ways in which

communication may occur, 1) Message Passing, 2) Broadcasting, and 3) Shared Memory. Because shared memory can cause scalability problems and broadcasting may produce excessive overhead, message passing appears to be the most economical and feasible approach. The fourth trait is in how agents model how other agents will perceive and act on the environment. The benefit of predicting behavior can assist in cooperation toward a common goal, and it is definitely possible in the NIM. A summary of the NIM using Cao's taxonomy is shown in Table 2-2.

Table 2-2: Cao's Taxonomy for the NIM.

Cao	**Centralization**	**Capabilities**	**Communication**	**Modeling**
NIM	Decentralized, Hierarchical	Homogeneous, without Bandwidth Limitations	Directed	Some

Dudek [9] proposed seven characteristics for multi-robot systems. Dudek's taxonomy is useful because it describes communication requirements more explicitly than Cao. The first characteristic is size, described as ALONE, PAIR, limited with respect to the environment (LIM-GROUP), and infinite (INF-GROUP). As each agent will reside on one node, the proper choice in the NIM is LIM-GROUP. The second characteristic is communication range. With COM-NONE, robots can not directly communicate, with COM-NEAR, they can communicate in a range sufficiently nearby, and in COM-INF they can communicate with any other robot. As networks gain size, efficient communication is generally limited to neighbors, thus the NIM is COM-NEAR. Communication Topology is the third characteristic in Dudek's taxonomy, dictating the

way that robots communicate. Communication can occur through broadcast only (TOP-BROAD), by name or address (TOP-ADD), by tree structure (TOP-TREE) or by graph structure (TOP-GRAPH). A communications network, being a graph, fits the TOP-GRAPH topology. The fourth characteristic, Communication Bandwidth, describes the cost and possibility of communication. The possibilities range from BAND-HIGH, where communication costs are negligible to BAND-ZERO, where communication is impossible. BAND-MOTION and BAND-LOW express intermediate cost incursion, but the communications network is best described as BAND-HIGH, although occasionally, as congestion rises, communication can approach BAND-ZERO. Reconfigurability, as the fifth characteristic, portrays the agents as fixed (ARR-STATIC), mobile (ARR-COMM) or rearrange arbitrarily (ARR-DYN). The node agents will likely be fixed in the scope of this work; however, future research could describe the nodes as being able to coordinate rearrangement to use resources better. Sixth, processing ability expresses how capable each agent is in regards to computation. This paper assumes that the processing power of an agent in the NIM is Turing Machine equivalent, the highest possible in Dudek's taxonomy. The final and seventh characteristic describes the composition of agents, homogeneous or heterogeneous. As in Cao's taxonomy, all agents are homogeneous in this domain. Table 2-3 summarizes how Dudek's taxonomy portrays the probable solution for the NIM domain.

Table 2-3: Dudek's Taxonomy for the NIM.

Dudek	**Size**	**Comm Range**	**Top-ology**	**Band-width**	**Reconfig-urability**	**Processing**	**Comp-osition**
NIM	LIM	NEAR	GRAPH	HIGH	STATIC	TME	Homo-geneous

By characterizing the NIM into the taxonomies of Cao and Dudek, this paper shows how implementing an agent based framework in the nodes of a network can be used to form a multi-agent system for network optimization. Hooper [16] used the taxonomies of Dudek and Cao to alter the three layer architecture to support communications requirements in multi-robot systems. His model, shown in Figure 2-5, is named HAMR (Hybrid Architecture for Multiple Robots), and introduces a coordinator component to work alongside the sequencing layer of the TLA. The coordinator is responsible for providing feedback to the Deliberator to aid in decision making, monitoring state, requesting new tasks if state changes significantly, and modeling other agents and the environment. According to Hooper, provided that modeling is extensive, communication can be significantly reduced using a coordinator. The addition of a coordinator element offers a simple construct for cooperation that can extend disparate agents using a three layer architecture into a cooperative system. Later, Hooper's design is adapted for a network agent using the unique taxonomy of a multi-agent system in the NIM problem.

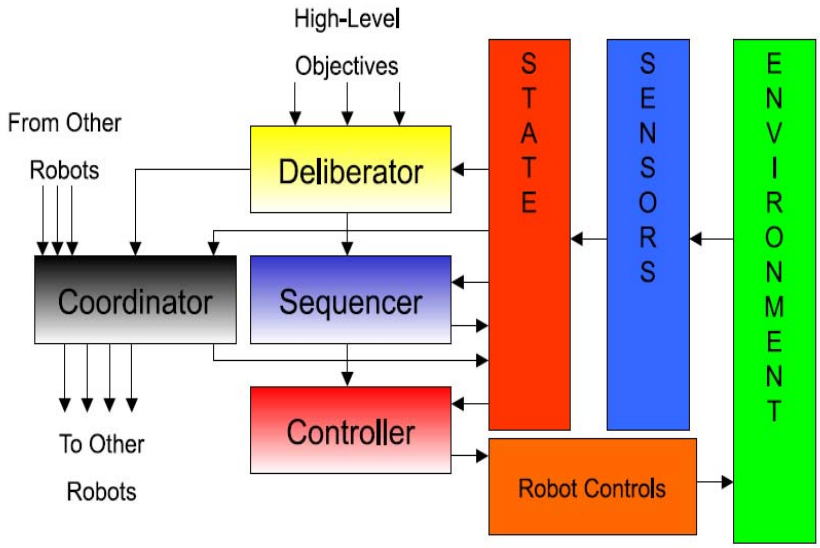

Figure 2-5: The HAMR Architecture Supporting Multi-Robot Systems [16].

Resource allocation in the network information maximization problem calls for a distributed agent that builds on reactive and hybrid robot control architectures. Using the modular UBF design, an agent is capable of dynamic and reactive behavior while allowing the system enough planning ability if new tasks need to completed. The characteristics expressed in the description of the NIM with Cao's or Dudek's taxonomy demands that some level of coordination occur between agents. The addition of a coordinator element is one such way to produce a cooperative multi-agent system. In order to use this agent-based framework, a communications node must support it. Next, this paper presents quality of service routing and middleware as the appropriate placement for such an agent.

Quality of Service for Information Maximization

Quality of Service is the ability to provide different priority to different applications, users, or data flows, or to guarantee a certain level of performance to a data flow [32]. As the introductory chapter describes, commanders must have assurances for important information in military environments. In addition, QoS guarantees increase network efficiency, which in turn increases the amount of useful information that can pass through a network. The ability to provide end-to-end quality of service is not the focus of this research, yet QoS guarantees are necessary for this research.

Many well known protocols exist that are able to route information based on an index, priority, QoS or other traffic characteristic. It is not the purpose of this work to build a new routing protocol. Instead, it asserts that for any network state, a routing protocol could perform better with adjustment. An illustration of such a scenario is described by Harmon [12]. Harmon stated that optimal TCP performance depends on its ability to estimate network performance and adapt. TCP estimates the condition of the network based on the reception of acknowledgments and perception of failed transmissions. In a challenged environment like wireless domains, failed transmissions may exist for a variety of reasons, yet TCP may attribute them to network congestion. In this situation, it fails to transmit when conditions are favorable, losing the opportunity and delaying successful delivery of the message. TCP can also result in suboptimal behavior when its timeout windows are not set appropriately in the wireless domain, causing excessive retransmissions and congesting the network with duplicate messages.

In this domain, TCP would perform better if its congestion avoidance mechanisms could be adjusted to fit the requirements of the wireless domain.

Likewise, the dynamic battlespace network environment demands a dynamic protocol. If, for any network state, there is a routing protocol which would perform as good as or better than any other, then a means for finding this routing protocol must exist. In this problem formulation, information maximization is the primary criteria of optimization, but the system is not complete unless it is able to efficiently use its resources. With the efficient use of resources, more information can get to the users who require it. At the same time, the problem of information overload may cause a commander to stop information at a certain utility from reaching select decision makers [1]. In this dynamic information environment, a dynamic means of selecting parameters for routing protocols is seen as essential to information maximization. To enable this concept, a local and automated system is presented in this thesis. For the purpose of assurance, it will be built on a routing protocol that supports QoS.

Protocol Support for QoS

A number of protocols can achieve reasonable guarantees for QoS. Resource ReSerVation Protocol (RSVP) [39] and Multi-Protocol Label Switching (MPLS) [33] are two protocols of many that do this by setting up reservations through network nodes. RSVP is a communications protocol that interfaces with a routing layer protocol to give stronger guarantees for end-to-end transmissions. It does this by scheduling reservations through a sink tree that reverses the list nodes that lead to the sink node (the return path) [39]. A data flow achieves QoS by three traffic control mechanisms, including (1) a

packet classifier, (2) admission control, and (3) a packet scheduler. The packet classifier determines the QoS class for each packet. For each outgoing interface, the packet scheduler or other link-layer-dependent mechanism achieves the promised QoS [4]. Admission control determines which reservations to grant and which to deny, maintaining a manageable network load [39].

RSVP operates in the following manner. RSVP establishes a sink tree that represents the reverse routes from each receiver to all source nodes. The network nodes maintain the sink tree using periodic path messages. Reservations are set up by propagating flow information from the receiver back to the source node. As each relay node in the return path accepts the reservation, it updates its state information to include the reservation. If any relay rejects the reservation, a reject message propagates back to the receiver. If a reservation is set up, reservation refresh messages are also passed among nodes to maintain reservations. When a packet is sent from a source node, the packet enters the routing layer of a relay at a classifier. The classifier determines a class of service for the packet and schedules it for transmission which includes checking for a reservation. When the reservation is no longer needed, the receiver sends a path teardown message.

The RSVP use of soft state to maintain reservations prevents the permanence of reservations that fail because of network errors. However, soft state reservations can use the network inefficiently in traditional implementations of RSVP because of their high communication overhead. The use of unreliable system messages can cause delays in

reservation setup and teardowns, leading to more inefficiency. As a result RSVP can have poor scalability and performance in dynamic networks [24].

MPLS addresses the scalability issue of RSVP. When packets enter a router, they are assigned a label by a label edge router. Label switched routers use these labels to forward packets along a label switched path (LSP). Network engineers design LSPs for a variety of purposes, but most importantly, for guaranteeing a class of service or routing around congested links. MPLS resides between the network layer and the data link layer. MPLS provides the mechanism for reserving a single path, using multicommodity flow algorithms to find an optimal set of paths given multiple flows. Among various types of MPLS protocols, several may split the flow along multiple paths of the network or keep them whole. Splitting flows may increase utilization but keeping them together keeps routing complexity low. [33]

The MPLS protocol is desirable because it can be used to guarantee a class of service to information flows and to perform dynamic routing tasks. Its widespread use and scalability properties are also desirable properties. Later, this paper presents a mechanism that reacts to the state of the network to perform routing tasks using a modified version of the MPLS protocol.

Middleware Support for QoS

Even with an underlying protocol, QoS requires some middleware support to ensure proper QoS objectives are met. While this work does not only support QoS, some work has been done in this field that relates to this thesis.

Like this work, Huang, et al. [19] describes a middleware service that assigns network resources to information flows based on their criticality. Huang analyzes multimedia flows into a critical, essential or non-essential class. This work generalizes the types of traffic that are analyzed by an information classifier, while Huang only considers multimedia traffic. Also, Huang focuses on scheduling algorithms to improve QoS, while this thesis takes a comprehensive view of the scheduling, classification and routing mechanisms.

In harmony with the opinion of this author, Kwiatkowski [22] states that when not enough capacity exists to send all information in military networks, messages carrying mission critical information should be delivered first. He argues that the traditional IP layer of networks is not capable of dynamically assigning information flows to network resources, because the network layer lacks an interface for dynamic resource allocation. Subsequently, Kwiatkowski builds a distributed software framework for network management at the middleware layer using three levels of interface. At the bottom level, right above the network layer, software abstractions of physical routing components provide a common interface for access. At the middle level, a service API provides the network management tasks of classifying and manipulating information flows according to their value. At the top level, a user interface interacts with a user or commander to set priorities to information flows. While this thesis is concerned with the same core problem and view that Kwiatkowski does, this work differs from his in many ways. One difference is the attributes used to assign value to information flows. In addition, he uses traffic engineering algorithms to manipulate flows, while this work asserts the use of

network agents. Finally, Kwiatkowski builds on a distributed processing environment using a CORBA based middleware framework, while this work will build on an agent based middleware framework. The key advantage of using an agent based middleware framework is simplicity because support decisions can be made close to the routing layer, while CORBA infers greater communication and processing requirements that affect speed or scalability.

Hopkinson [17] created an agent based middleware framework to synchronize various components of a power system simulator. One of his components is in NS2 [10], a network simulator developed by the University of California at Berkeley, Lawrence Berkeley Labs, the University of Southern California, and Xerox PARC. NS2 is a high-quality simulator, which allows the creation of a wide variety of communication scenarios. NS2 is able to simulate the behavior of network protocols under various forms of stress, such as might be caused by competition for network resources when multiple applications share a network and communicate over the same routers and communication links. An agent in every communications node inside NS2 coordinates with other nodes using system messages. While the framework was not created to support QoS, it was extended to do this by Llewellyn [23]. Llewellyn used a reservation based routing protocol and supported its QoS properties by establishing fault recovery mechanisms. This work extends the work of Llewellyn and Hopkinson, using the agent based middleware framework and reservation based routing protocol to establish underlying QoS guarantees. Then it applies information centric resource allocation to support QoS routing using techniques that are described later.

QoS guarantees are critical to support network information maximization. Reservation based routing protocols like MPLS provide a base for QoS in a network. Middleware support can increase their effectiveness and provide a foundation for other optimizations.

Research Overview

By using the underlying support of a QoS mechanism, this research is aimed at extending a middleware framework in each network node that will maximize the utility of information delivered by a network. Assuming classification and scheduling mechanisms are in place; this agent based framework will allocate network resources, namely bandwidth, in a MAS to produce valuable information to a user.

Summary

The problems of information classification, scheduling, providing quality of service, and resource allocation converge in the network information maximization problem. Information classification segments information on three attributes: mission association, timeliness and user input. Scheduling combines these attributes into an information utility that acts as an index to solve the information flow scheduling problem, a special case of the Restless Bandit Problem. Assuming these problems can be reasonably solved, an agent employing robot control architecture may allocate bandwidth and other resources to maximize the aggregate utility of information flows throughout the network. Additionally, using a QoS framework, end-to-end transmission is guaranteed. The following chapter presents an information maximization agent, using information classification, scheduling, robot control and QoS mechanisms.

III. Methodology

As discussed in Chapter I, the long range vision of this work is to create a comprehensive system for ensuring information is optimally gathered from a network. The comprehensive system would need to contain an information classification mechanism, scheduler and a resource allocation and routing scheme.

Chapter II describes different information classification approaches that indicated the information classification mechanism's design. In it, a commander could interface with an application to create an NTO to articulate mission objectives and their priorities. The interface would deliver mission goals to a dynamic GMUGLE type goal decomposer [13], breaking high level goals into low level goals and tasks, each weighted with respect to how well they fulfill mission requirements. Another application distributes sub goals and tasks to individual nodes. Individual nodes would contain a classification module to characterize an information flow along three attributes: mission association, timeliness and user input. A scheduler assigns an expected utility for the current state of the information flow, and the information flows could be greedily selected to find a near optimal set of traffic in a discrete time window.

Given that there is a means of assigning such a utility, and scheduling messages based on this, there would need to be a means of routing these messages to their end destination. A QoS support protocol running on top of IP could give some guarantees in routing, but since reservations are involved, the protocol requires a means of distributing

bandwidth resources. The purpose of this chapter is to describe a methodology to create this part of the system, the resource allocation mechanism to work with QoS routing.

Domain

The information system will reside on a hybrid communications network with wired and wireless links. Such a network contains long lasting connections and ones that are intermittent. Network nodes can have highly complex interactions, some may communicate over a multitude of duplex links, and others may communicate over single, simplex links. Without belaboring the point, a hybrid communications network can be highly complex; therefore for the purpose of introducing the information system, a smaller network is described for initial work.

In this example network, a single sink is buffered to the network by a single relay. The relay draws information through a network of relays. In this particular example, there are three relays. Two nodes, performing information harvesting, connect to each of the relays. Figure 3-1 depicts this network. Initially, the following assumptions are made. First, the network has a hierarchical architecture, meaning that exterior nodes gather data from the environment, while interior nodes have processing power and perform routing tasks. Second, nodes have a uni-directional data flow, so information travels from source to sink. However, acknowledgements and service messages can travel from sink to source. Further, the network is setup using an end to end QoS protocol on top of IP. MPLS and other QoS protocols use reservations to set a path through a network.

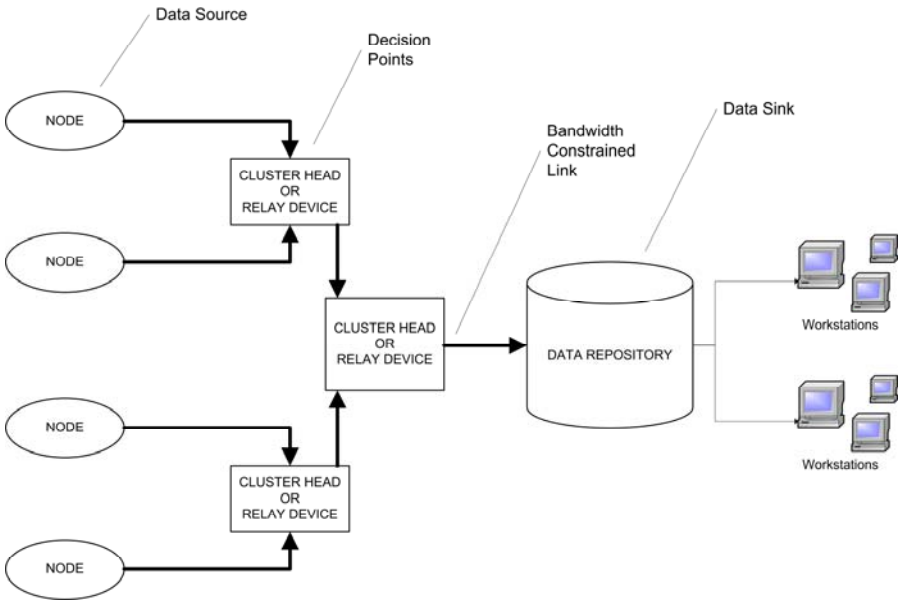

Figure 3-1: Simple Network.

Information flows that do not have a reservation are sent as best effort traffic. Best effort transmission is analogous to flying standby instead of with an assigned seat. A reservation policy sets up how much bandwidth of the total available is used for reservations and how much is used for best effort. Once reservations are filled, the remaining messages are all sent best effort. Deciding how bandwidth is assigned over time to fill these reservations is the concern of the resource allocator.

Requirements

As stated in Chapter I, the system needs to fulfill five requirements. Here, these are reviewed in light of the domain, giving an indication of the solution to follow.

Requirement 1 – Create a network that makes 'thoughtful' decisions on what traffic to drop and what traffic to keep in relation to mission importance.

First and foremost, this network must route high utility information to the sink. If there were no bandwidth constraints, this task would be simple, since all traffic would be routed to the sink. If the network is limited on the links between the nodes and sink in bandwidth, then agents in the network must decide what information should flow to the sink and what should not. Because of the assumed existence of a classifier module, finding high utility information is simpler. The classifier module determines each packet's utility when it enters the system by associating the packet with a known information flow. The optimal set of traffic to send to the source includes information flows with utility above a certain threshold.

Requirement 2 – Create a network that balances network resources and mission objectives.

Relays must choose an optimal traffic set to the source. The optimal set of traffic not only includes the traffic with highest relevance, but also whatever information flows may fit in the bandwidth constrained channel's remaining capacity after the top flows are assigned. Leftover bandwidth makes this problem more complicated, because it allows for a situation in which the information utilities of two flows are similar but their sizes are different. If the higher value one is also smaller and leaves a greater gap in the channel, it may be more appropriate to send the slightly lower value stream if it increases utilization. Also, the agents in a network must be able to decide when it is appropriate to squelch on information flow so an alternate flow may proceed.

Our network model naturally accounts for this. A certain portion of the network is allocated for reservation only traffic, while other traffic is sent best-effort, or as space becomes available. Balancing network resources and mission objectives relies upon balancing how much bandwidth is set aside for reservations and for best effort traffic as well as deciding how to manage reserved traffic.

Requirement 3 – Create a network that makes distributed decisions about what information needs to flow from source to sink.

A network that makes distributed decisions is one in which global knowledge, contact, and control is not available. In this design, each network relay takes action in the environment only using information gained through perception and peer communication. Peer communication is limited to immediate neighbors, but does not limit the amount of information that can be shared. Such a network may be thought to have individual agents at each node, which have the means to communicate with agents at neighbor nodes, using shared links between them. Further, each agent will be required to take action based on its perception of the environment and this shared information.

Requirement 4 – The network must adapt to periods of bandwidth flux.

The existence of communication between nodes allows neighbor nodes to become aware when their neighbors fail or when links degrade. This information must propagate quickly in the network so nodes using these failed or degrade links can react to changes and route critical information through different routes. When bandwidth flux is significant enough, affected nodes may be forced to recalculate entire routing tables and reservation assignments.

Requirement 5 – Create a network that negotiates multiple, competing mission objectives.

Scaling the network to handle multiple mission objectives requires a change in the way a goal decomposer/task distributer works. Instead of sending out task weightings from one source, having multiple sources forces arbitration between competing mission objectives at some level in the system. This work abstracts this point, since it is not concerned with the details of the information classification process.

However, the network must be able to route to multiple sinks in the network. The hierarchical assumption made earlier precludes that source from being on the interior of the network, because it alters the capabilities of interior nodes. The uni-directional flow assumption also precludes the interior node because it may cause the flow of information to go from source to sink to source. These assumptions have been made to simplify the analysis of the network's performance, though it may be possible that the design will scale to situations like this as well. In this work, a multiple sink test is devised to illustrate this requirement.

Conceptual Design

Assuming that information classification techniques can produce an information utility index based in part on mission objectives, it is reasonable to tailor a system around this index to manage the information flow in a network. Using a utility threshold within each node is an effective way to manage indices to fulfill the requirements. Responding to the requirements of having a distributed and adaptable system is achievable through

employing network agents based on robot control architectures. This section discusses the concepts of utility threshold management and an agent based framework in further detail.

Utility Threshold

The requirements point out two potential decision making points, as shown in Figure 3-2: 1) whether or not to send or hold in a queue, based on a threshold, and 2) whether or not to make a reservation, based on available reservation space. A utility threshold is merely a number assigned to eliminate traffic overload. Used simply, messages above the thresholds are sent, messages that are below, are not. Thresholds could also be used to set up a reservation. Available reservation space is the percentage of the average bandwidth capacity to be used for reservations. If no reservation space exists, reservations can not be made. The remainder of the average bandwidth capacity, equal to the total capacity minus the available reservation space, is used for best effort traffic flows, those which could not gain a reservation. The basis for setting up a reservation is partially dependent on its utility as it compares to the utility threshold, since a reservation won't be made for information flows that do not meet the threshold.

Thus, thresholds become the main information management device in the system. Packets are not generated unless they meet the threshold, and they are not routed with reservations if a downstream node has a higher threshold to meet. During the process of sending packets with these information flows, some nodes may become overwhelmed, arbitrarily dropping packets. Therefore, packets that do not meet the nodes threshold may lose their reservation while higher value packets persist.

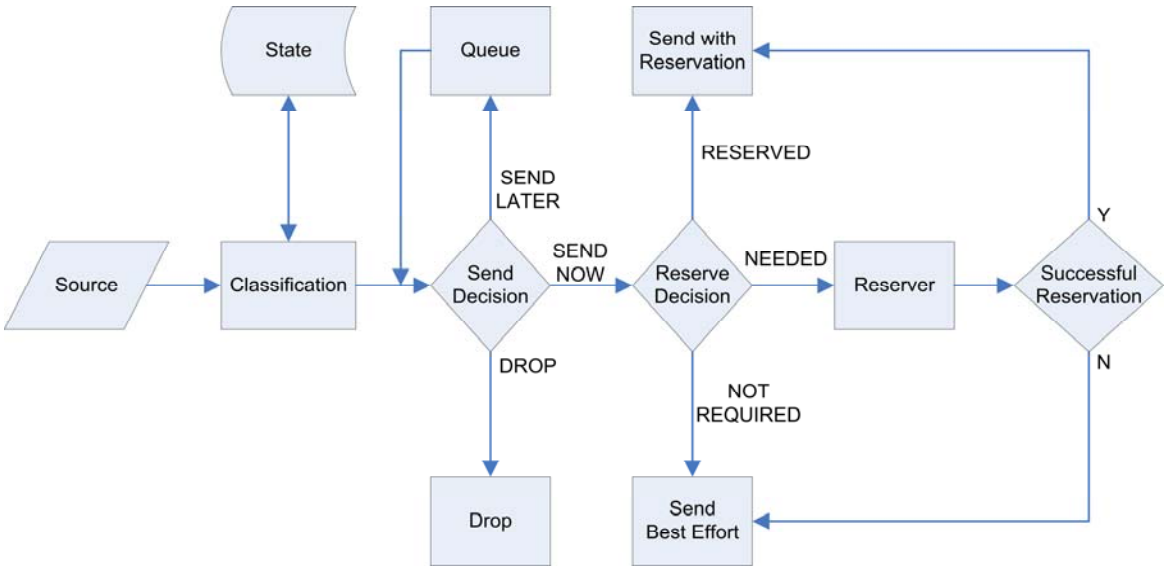

Figure 3-2: Generalized Routing Decision.

Managing the utility thresholds becomes a crucial capability. Keeping a distributed architecture in mind, agents must synchronize their threshold management to use resources effectively, becoming highly reactive to their neighbors, so changes can propagate rapidly. One means of synchronization is through sending system messages to neighbor nodes to either raise thresholds when links become congested, or lower them when links are underutilized.

Using this method, the system maintains the distributed requirement stated earlier. Further, utility thresholds perform well to adapt to bandwidth flux. When links degrade, thresholds will naturally increase on those links. When links fail, nodes notice their absence when coordination messages are no longer received by neighbor nodes.

The Hybrid Agent for Network Control (HANC)

Provided that utility thresholds will be able to fulfill the requirements, something must be able to perform the management functions on the threshold. Applied at every node, the three layer architecture in multi robot domains can fit well with the needs of this domain. It has the capacity to be reactive to the state of the network, while it can also be tailored to the demands of the mission. Thus, agents employing the three layer architecture are applied here to select the actions of managing thresholds, sending raise and lower messages, and managing reservations. This section describes HANC, the Hybrid Agent for Network Control.

The HANC control architecture consists of three layers of control, as well as two additional components for coordination and state information. The three layer, two component construct is illustrated in Figure 3-3. The first and highest layer, the deliberator, is a pure planner. It decides what the best course of action is for a particular long term goal. For the sake of this work, we will consider the commander or the person carrying out commander's intent to be the deliberator. Based on his view of the network, he may opt for greater throughput or utilization in the network at the cost of utility. He may want to lower the amount of information coming into the battle center or increase security and he communicates his intent through a network tasking order.

Deliberator input from an NTO gains access to the agent through a coordinator component, which passes this input to a sequencer layer. An interface to receive the

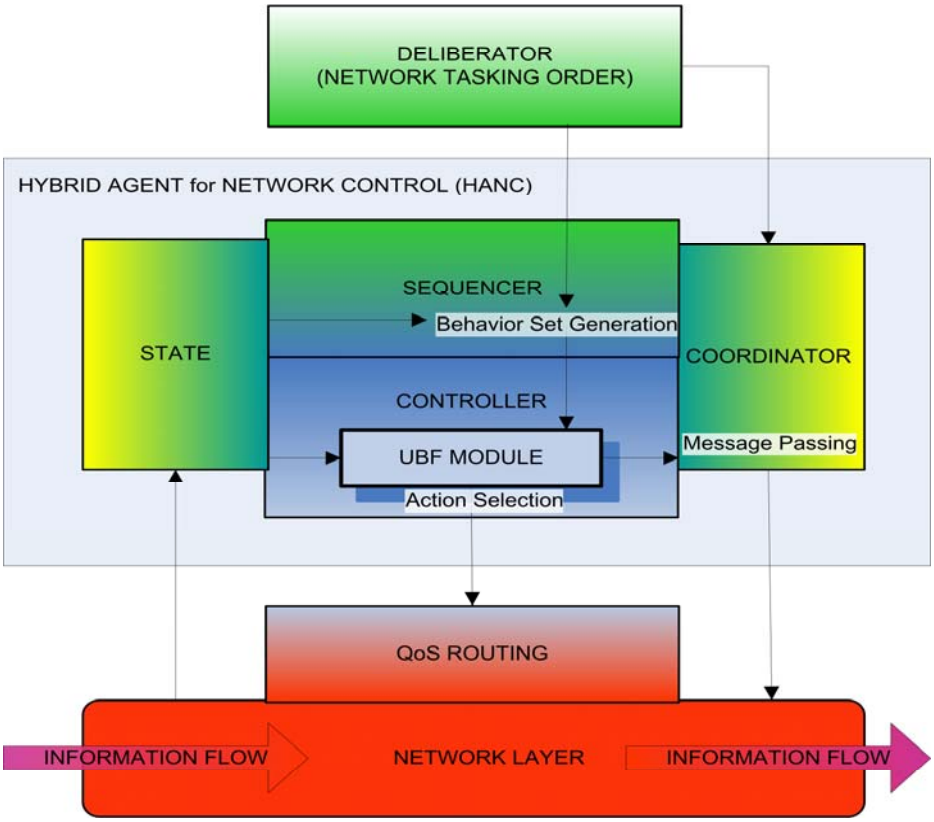

Figure 3-3: Conceptual Design of HANC. HANC contains a controller layer that generates an action on the environment through the UBF. The sequencer reacts to the deliberator by creating a set of behaviors that will optimize the network to the Network Tasking Order. A coordinator module is necessary to maintain coordination with other agents through message passing.

network tasking order at the sequencer layer constitutes the functionality of a deliberator. The deliberator interacts with a sequencer to carry out long range planning.

The second layer is the sequencer, which is used to weight the tasks at the layer below based on the goals being generated from the deliberator. Two of the concepts covered thus far have been maximizing utilization and minimizing information overload. The two are obviously divergent goals, the former calls for all information to be sent to the user, while the latter calls for a strict adherence to only the amount of information that

a user can absorb. The sequencer may use an expert system, machine learning, or manual input to alter how the controller layer acts. Before illustrating how that interaction occurs, it is necessary to discuss the next layer of control.

The third, lowest layer is the controller, the reactive layer. In a purely reactive environment, the controller hardwires sensor input to actuators performing actions in the environment. To increase the flexibility of a pure reactive and make it act more intelligently, sensor input is buffered into memory and is stored as state. From the current state, agents choose how to act. In this design, the UBF is the controller layer of the system. The UBF is useful because it simplifies development and testing, promotes code reuse, allows complex behaviors to emerge from basic ones, and allows system developers to use the architecture they feel works best [37].

Using a UBF at the controller layer, the sequencer in this system selects among a library of all behaviors to create behavior sets that can fit the proper system objective. It also selects an arbitration unit for the UBF to select or combine an action from all the actions that the set of behaviors generate. In Figure 3-4, a network is perceived by the agent and perception is stored in state. The composite behavior, in the upper left of the figure, is populated with behaviors and an arbiter by the sequencer layer. Next, the behaviors generate candidate actions for the composite behavior based on the current perceived state of the node. Then, the candidate actions are sent to an arbiter to create a composite action. The HANC Agent executes the composite action.

In this work, the sequencer is manually adjusted, based on the implementer's expert knowledge about the network. In future work, the sequencer can be extended to

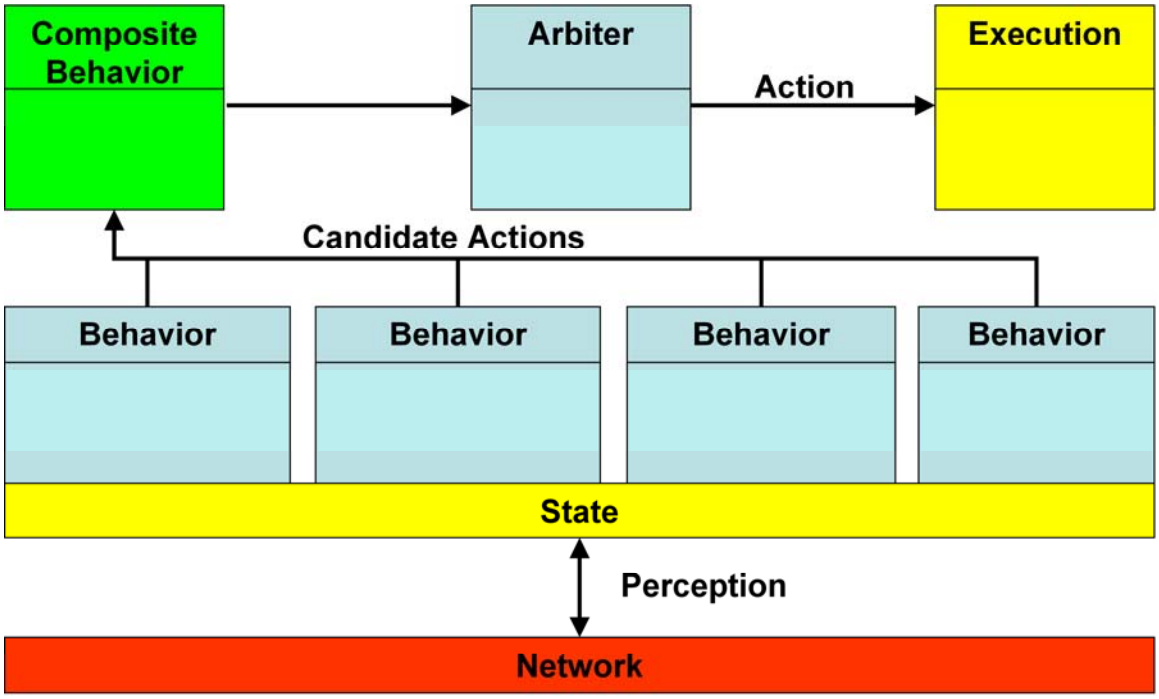

Figure 3-4: Conceptual Design of the UBF Module. The UBF Module resides in the controller layer in Figure 3.3. The sequencer layer populates the composite behavior with and arbiter and behaviors to create an executable action based on perceived network state.

accept NTO input and weigh this input against its state information. State is the second component in the HANC agent. It stores the agent's perception of the network and some limited trend analysis of network traffic characteristics.

To summarize, HANC, as depicted in Figure 3-3, directs information flow for its resident node. In this system, the controller and sequencer layers reside on every node while the deliberator is abstracted to the end user. The controller is supported by the UBF which creates a policy for threshold and reservations to determine how the router will run. Shown in Figure 3-4, data enters the nodes and feed into the sensors and state of HANC. Goals from the deliberator are issued to the sequencer to help it select a set of behaviors and an arbiter for the controller layer. The behaviors generate actions that are

selected or combined in an arbiter. The selected or combined action is executed in the environment.

Specific Design

HANC can be used for many optimization tasks, but to show its capability in the information maximization problem, it is limited here to a subset of its possible uses. In this section, the agent is specifically designed to maximize information utility, outlining how the sequencer interacts with the controller and how the controller reacts to perception to produce actions. Starting with perception, the specific design explains the UML diagram in Figure 3-5.

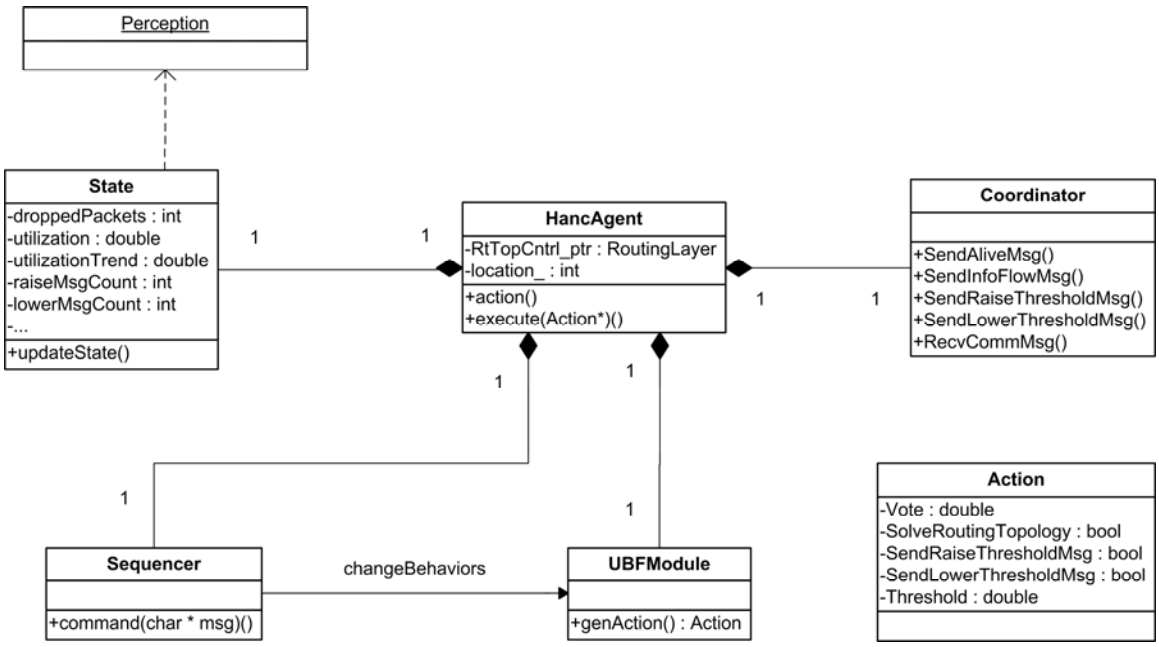

Figure 3-5: UML Diagram for HANC. HANC is supported by four components: State, Sequencer, Coordinator and UBF Module. HANC interacts in the network environment by generating and executing actions.

Perception

In the physical robot environment, a pure reactive robot is one in which sensors are directly attached to actuators so that sensing produces near immediate action. Although the UBF uses a limited amount of state memory to store sensory input to generate actions it still remains reactive. The hope for this work is to maintain the reactive property of the UBF to make the system dynamic. The key to plugging the UBF into to network domain is in finding meaningful preceptors and tying perceived events to actionable parts of the system.

Three meaningful sources of perception stand out. Inbound data packets are the first source. Inbound packet header information includes flow identifiers, utilities, and size. The rate of inbound traffic can also indicate what is occurring at upstream nodes. System messages are a second source. As nodes get overwhelmed, they generate raise threshold messages. Conversely, when nodes are not being appropriately utilized, lower threshold messages are sent. Both types of messages can be used to perceive the environment. Internal node characteristics are a third source of perception. Nodes can perceive their own queue sizes, outbound data rates and routing tables. To be able to apply these perceptions, they are stored in state.

State

To tie perception to action, HANC must maintain state information. State contains all the perception elements and can also be altered to monitor trends in the data. With some decay rate, state information can be removed or replaced as it loses meaning or certainty. For instance, state may attempt to estimate the utilization in a link by taking the

average amount of traffic sent over a time period divided by the average amount of bandwidth available in that same time period. In time, however, this metric becomes inaccurate to current network characteristics and must be recalculated.

In this design, shown in Figure 3-5, some of the state variables are the number of packets dropped in the previous second (droppedPackets) and the current utilization of the outbound queue (utilization). Averaging utilization over time, another variable is established, utilizationTrend. State also maintains the number of raise and lower threshold messages received by the coordinator component. To update state information, HANC calls an update state function. Figure 3-6 shows the pseudo code that places this command in HANC's execution function, named action( ).

```
void HANCAGENT->action( ) {        //main  call of HancAgent, by AgentHQ
    //Update state variables for Sequencer and UBF_Module use
    State->updateState( );
    //Send a command to the sequencer to decide behavior sets for the controller
    Sequencer->command( );          //calls UBF_Module->changeBehavior( )
    //Execute the action returned by the call to the UBF_Module
    execute( UBF_Module->genAction( ) );
    //Send ping to neighbor nodes that implies that this node is still functioning
    Coordinator->sendAliveMessage( );
}
```

Figure 3-6: HANC's Action Function. The main call of the HANC Agent updates state information, calls the sequencer module to build behaviors in the UBF, and next calls the UBF Module to generate an action. The execute function performs the action and sends an "alive" message through the coordinator.

Sequencer

As proposed in the previous section, the sequencer layer of HANC must be able to choose between maximizing utilization and minimizing information overload based on the input given from the deliberator level. The sequencer can do this by selecting a different set of behavior when one of these optimizations is called for. The behaviors that propose to maximize utilization are described below. The sequencer employed by HANC in this iteration of research is rudimentary, consisting of an interface that must be altered manually. However, the sequencer in this design can be called with command to change the behavior set in the UBF Module.

Coordinator

Because the UBF Module and execute(Action) function use the coordinator, the coordinator is described before the lowest layer of this design. The coordinator component of HANC is responsible for sending and receiving coordination messages from other HANC agents. In an expanded system, a part of this responsibility is the dissemination of the NTO. At the sequencer layer, this coordination is concerned with having fellow agents optimizing for the same network tasks. At the controller layer, which this paper focuses on, coordination involves sending messages to neighbor nodes to accomplish a task. For example, the coordinator periodically sends "alive" messages to its neighbors to inform them that the link between two nodes is functioning. A loss of this alive message would cause a solve routing action to be executed. The UBF Module has a subset of actions that involve leveraging the coordinator component. These actions and others are described next.

Actions

In the UBF implemented by Woolley [37], each behavior in the behavior set accesses state information and generates an action and a vote. The action is the behaviors best guess on what to do and the vote can be thought of as a confidence that the behavior should be implemented. Individual behaviors can choose to generate a single action or many, depending on how many degrees of freedom are available to the agent. The four actions shown in Figure 3-6 are described below. Figure 3-7 illustrates how HANC's execute(Action) function performs these tasks.

Threshold – The threshold value is based on perceived demand at downstream sources, ultimately the sink node, propagating the user's preference and mission demands throughout the system. HANC manages the threshold value by detecting threshold demands at other nodes. Upon the receipt of raise threshold messages and lower threshold messages, the agent acts accordingly to select a new threshold. The threshold value is used to generate information flows and also to maintain reservations. If a reserved flow passes through a node requiring a higher threshold, the reservation is removed.

Send Raise Threshold Message - HANC is also responsible for generating actions that would help its neighbors manage threshold values. For this action, the UBF Module leverages the coordinator's send and receive mechanisms. The agent could choose to send raise threshold messages based on its perception that too much information is coming in. One way to do this is to monitor queue sizes. As packets drop from queues, it is an indication that too much traffic is coming to the node. A count is updated in state that maintains the current amount of lost packets. Proportionally, the vote to send a raise

```
void HANCAGENT->execute(ACTION action ) {       //called by  HANC Agent
    /***************ADJUST THRESHOLD ***********************/
    if ( action->getThreshold( ) > currentThreshold )
        State->decreaseRaiseCount( );  //decrease raise threshold message count
    else
        State->decreaseLowerCount( ); //decrease lower threshold message count
    currentThreshold = action->getThreshold( )
    /***************GENERATE INFO FLOW MESSAGE*************/
    //If this node is a source node for information flows...
    if (InfoFlow->Utility > currentThreshold)
        Coordinator->sendInfoFlowMessage(InfoFlow->ID);
    /***************SEND RAISE THRESHOLD MESSAGE***********/
    if (action->getSendRaiseThresholdMsg( ) )
        for each node in UpstreamNodes{    //send to upstream neighbors
            Coordinator->sendRaiseThresholdMessage(location, n->location);
        }
    /***************SEND LOWER THRESHOLD MESSAGE**********/
    if (action->getSendLowerThresholdMsg( ) )
        for each node in UpstreamNodes{    //send to upstream neighbors
            Coordinator->sendLowerThresholdMessage(location, n->location);
        }
    /***************SOLVE ROUTING TOPOLOGY*****************/
    if (action->getSolveRoutingTopology( ) )
        RoutingLayer->solve_routing( );
}
```

Figure 3-7: HANC's Execute Function. The pseudo code describes four actions that are taken when the proper parameters are set by the UBF Module. The first four actions focus on information utility, while the fifth action focuses on adjusting to bandwidth flux.

threshold message increases. Packet drops are counted until enough packet drops will force a message to send or enough time has passed that the entire packet drop count is not meaningful.

Send Lower Threshold Message – Likewise, HANC could send a lower threshold message based on its perception that too little information is coming in. Using queue sizes again, if queues are empty or near empty, it is an indication that too little traffic is coming to the node. Thus, the vote to send a lower threshold message increases, and the same process of sending a raise threshold message applies. Again, this action is executed using the coordinator's send and receive mechanisms.

Solve Routing Topology – As stated earlier, the coordinator sends "alive" messages periodically to its neighbors to inform them that their shared link is still fit for use. When messages cease to cross a link, the neighbor makes the assumption that the link is no longer valid. To compensate, a solve routing topology action is generated. This action, when executed in the HANC agent, replaces the old routing table and all reservations for information flows with a new configuration.

UBF Module

Having described the actions generated by the controller layer, the UBF Module is presented. The UBF operates in a hierarchical fashion to proliferate code reuse. Figure 3-8 presents the structure of the UBF in a UML class diagram. The major components of the design are arbiters and behaviors. The goal of this structure is to generate actions from simple behaviors which view only a portion of the state to limit complexity and increase speed.

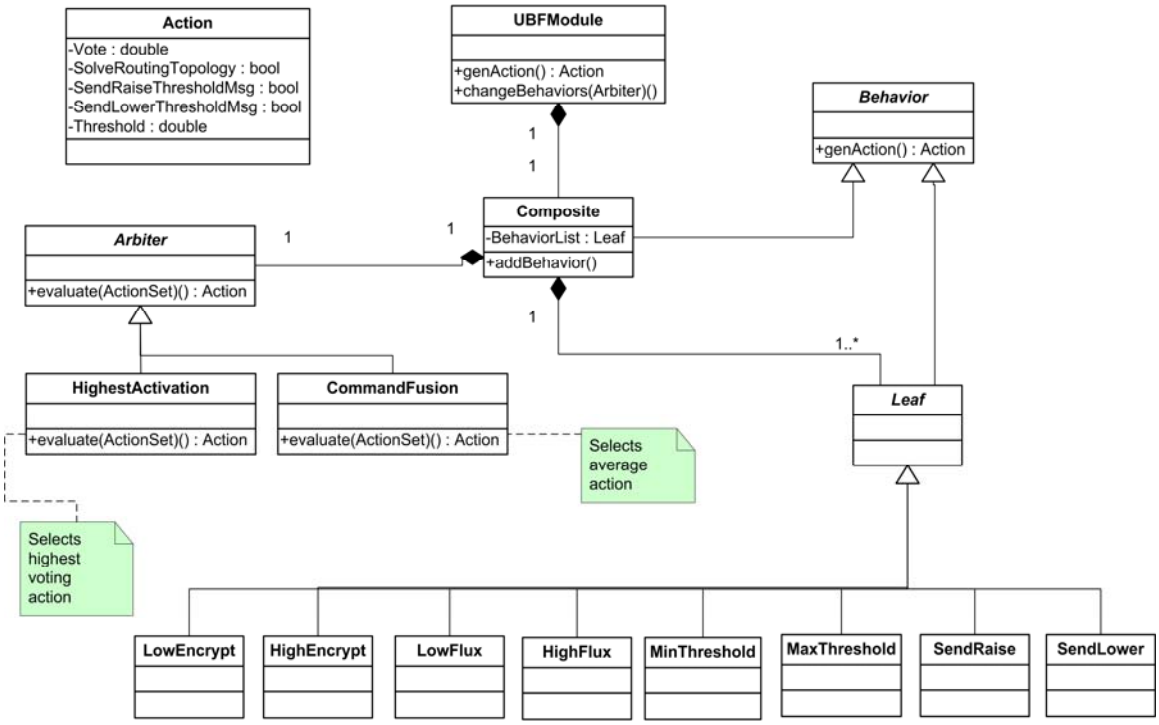

Figure 3-8: UML Diagram for the UBF Module. The UBF's composite behavior consists of an Arbiter and several Leaf Behaviors. The Leaf Behaviors generate candidate actions that the arbiter combines into a single action. The action is returned to the Agent.

The UBF Module has two main functions that handle operations. The first was mentioned in the description of the sequencer. When the sequencer decides to change the task of a node, it chooses a behavior set that may be different from the previous set. The sequencer accesses UBF_Module->changeBehavior(Arbiter) to set the new behaviors and arbiter in the Composite Behavior class. The second function is the main execution of the UBF Module which calls the Composite->genAction( ) function to generate an action. These two functions are written in pseudo code in Figure 3-9.

```
void UBF_MODULE->changeBehaviors(ARBITER arbiter){ //called by Sequencer
    Composite->setBehaviorList(new BehaviorList);
    for each LeafBehavior of all BEHAVIORS {
        if ( LeafBehavior->isActive( ) )
            Composite->add(LeafBehavior);
    }
    Composite->setArbiter(arbiter);
}

Action UBF_MODULE->genAction( ){      //when called by HANCAgent
    return Composite->genAction( );
}
```

Figure 3-9: UBF Module Functions. In changeBehaviors(Arbiter), the Sequencer adapts to a task by choosing appropriate behaviors for the Composite behavior. In genAction( ), the Composite behavior generates an action for the HANCAgent.

The UBF Module tasks the Composite class to generate the action for the HANC agent. In turn, the Composite class's genAction( ) develops an action by calling on one or many Leaf Behaviors to perform their genAction( ) functions. The resultant actions are gathered in an ActionSet for later arbitration. Once all Leaf Behaviors have the chance to add an action, the ActionSet is passed to the Arbiter for evaluation. A representation of the pseudo code is shown in Figure 3-10. The next subsection describes a selection of the Composite class's Leaf Behaviors and Arbiters for use in the Network Information Maximization problem.

```
Action COMPOSITE->genAction( ){          //when called by UBF_Module
    ACTION action = new ACTION;          //action to be returned
    for each LeafBehavior in  LeafBehaviorList {    //iterate all behaviors
        //each Behavior generates an action
        ActionSet->add(LeafBehavior->genAction( ));
    }
    action = Arbiter->evaluate(ActionSet);   //arbiter selects/combines actions
    return action;
}
```

Figure 3-10: The Composite Class genAction( ) Function. Leaf Behaviors offer actions that are selected or combined in the Arbiter's evaluate function.

Behaviors

Leaf Behaviors, described above, implement the genAction( ) function of the Leaf and Behavior abstract classes. They do so by focusing on generating particular actions, created with a limited view of the state space. While several behaviors may be able to work inside the network domain, three pairs are discussed here that specifically address the NIM. The first pair, MinThreshold and MaxThreshold, seeks to set threshold to 0 and infinity, respectively. The normal vote for MinThreshold is to vote decay the current threshold by one percent, so there is never a situation where some information stream is never sent while bandwidth is available to do so. When MinThreshold receives a lower threshold message, it votes higher to lower the threshold by a greater amount. Conversely, MaxThreshold raises the threshold after receiving a number of raise threshold messages. Figures 3-11 and 3-12 show the pseudo code for these behaviors.

```
Action MAXTHRESHOLD genAction( ) {                    //called by Composite behavior
        ACTION action = new ACTION;
        //Set the threshold to a decayed value of the last threshold
        double threshold = State->getLastThreshold( );
        //Get the number of Raise Threshold Messages received
        int RaiseCount = State->get_received_Raise_Threshold_Messages( );
        //Set the threshold to account for these Raise Threshold requests
        threshold = threshold + RaiseCount;
        action->setThreshold(threshold);              //Set the action attributes
        action->setVote(1);                           //Assign a vote
        return action;                                //Return the action
}
```

Figure 3-11: MaxThreshold genAction( ). Seeks a high threshold.

```
Action MINTHRESHOLD genAction( ) {                    //called by Composite behavior
        ACTION action = new ACTION;
        double decay_rate = .992;                     //possible parameter
        //Set the threshold to a decayed value of the last threshold
        double threshold = State->getLastThreshold( ) * decay_rate;
        //Get the number of Lower Threshold Messages received
        int LowCount = State->get_received_Lower_Threshold_Messages( );
        //Set the threshold to account for these Lower Threshold requests
        threshold = threshold – LowCount;
        action->setThreshold(threshold);              //Set the action attributes
        action->setVote(100);                         //Assign a vote
        return action;                                //Return the action
}
```

Figure 3-12: MinThreshold genAction( ). Seeks a low threshold.

A second pair of behaviors, SendRaise and SendLower, is created in a similar way. SendRaise seeks to send raise threshold messages when dropped packets appear, while never seeking to send a lower threshold message. The SendLower behavior seeks to send lower threshold messages when the state contains information that the queues are empty. Another behavior pair is HighFlux and LowFlux. These two behaviors adapt to the fluctuations in bandwidth on the network, generating an action to solve for the routing topology. HighFlux is coded to expect a network with a lot of nodes going in and out of the network, and likewise waits a longer time to reroute information because the failed node in question is expected to come back soon. LowFlux makes the assumption that when links are down, they go down for good. Therefore, LowFlux reacts aggressively to the loss of a network link or node. The pseudo code for these behaviors follow the pattern of the MaxThreshold and MinThreshold behaviors, but they are contained in appendix A for further study. For optimizations outside the scope of the NIM, additional behaviors may be needed. Regardless of the behavior set, an arbiter is needed to select or combine all generated actions.

Arbiter

The UBF can use a number of arbiters to select an action from many behaviors. A Highest Activation arbiter chooses the action with the highest vote. A Command Fusion arbiter, described in Figure 3-13, selects an average action for all actions in the ActionSet. The Command Fusion arbiter in this design takes the MaxThreshold and MinThreshold behaviors and sets a threshold that is the vote weighted average between the two. It sets SendRaiseThresholdMessage to true if the number of votes to send raise

```
Action COMMANDFUSION->evaluate(vector ActionSet ) {  //called by  Composite
    ACTION arbitrated_action = new ACTION; //the combined action to be returned
    int votes = 0;                                  //temp vote tally variable
    double sumThreshold = 0;
/*******************THRESHOLD ARBITRATION****************/
    for each action in ActionSet {              //iterate through all actions
        if (action->isThresholdSet( ) ) {       //if this attribute is set...
            votes += action->getVote( );                //increase the votes
            //increase the vote weighted sum
            sumThreshold += action->getVote( ) * action->getThreshold( );
    }   }
    //Set the action to the average sum
    arbitrated_action->set(sumThreshold / votes);
    votes = 0;                                  //reset votes for the next attribute
/**********SEND RAISE THRESHOLD MESSAGE ARBITRATION******/
    for each action in ActionSet {
        if (action->isSendRaiseThresholdMessageSet( ) )
            votes += action->getVote( );        //increase votes when set
        else  votes -= action->getVote( );      //decrease votes when not set
    }
    // If votes for this were more than votes against this set to true
    arbitrated_action->setSendRaiseThresholdMessage (votes > 0);
    votes = 0;
/****************Other actions set in similar fashion*********************/
    return arbitrated_action; //returns the average of all actions
}
```

Figure 3-13: The CommandFusion Arbiter Evaluate Function. An arbitrated action is returned to the Composite function that is a vote weighted average of all actions.

threshold messages exceeds the number of votes not to. It does the same for send lower threshold message and solve routing topology. The exact point chosen between the behaviors is a matter of expert knowledge or trial and error to set how individual behaviors vote. In this instance, trial and error is the preferred method of finding the good balance between the behaviors.

Expected Results

The system designed in this chapter should meet the requirements that were devised in the introductory chapter. To show this, a series of tests are described to evaluate whether the system has performed up to its intended standards.

Test 1 – Arbitrary Drop Test

If the network is limited in bandwidth capacity and as a result, can not carry the entire traffic load for any particular time step, traffic will be dropped with little regard to its effect on average utility at the sink node. As a default, network queues are set up so to enque higher utility packets while dequeing and dropping lower utility packets. What the queues can not do, however, is perform a system wide adjustment to the information flow to maximize the average utility at the sink. To pass the arbitrary drop test, HANC must be able to improve the system wide aggregation of data. This will be done by analyzing received traffic at the sink node. If average utility is higher using the multi agent system, then the arbitrary drop test is passed.

Test 2 – Utilization Test

In a default setting, an overloaded network will utilize bandwidth close to the network's peak rate. In adding the multi agent system, network utilization, must not fall

from this peak rate. The first part of the utilization test is to ensure that equivalency holds between the agent framework and the default mode on the network's overloaded links. This is the equivalency condition.

As portrayed in Figure 2-3, a situation that involves alternate flows utilizing the same links as primary traffic may force alternate links to be underutilized. The second part of the utilization test is passed if utilization is improved on the network's underutilized links. This is the improvement condition. Therefore, the utilization test is passed when the equivalency and improvement conditions are met.

Test 3 – Bandwidth Flux Test

When a routing protocol using reservations is employed, the ability of the reservations to be rerouted to links that work is critical. The requirement that drives the bandwidth flux test called for a system that could adapt to link degradation or failure. If the system were not employed, we would expect the default behavior to drop traffic until human intervention corrected the fault. To prove that HANC can adapt to periods of bandwidth flux, it must demonstrate the ability to recognize and recovery from link failures. It also must demonstrate the ability to cope with new links in the network. A scenario that shows a network node failing then later restoring can show these capabilities.

Test 4 – Multiple Sink Test

When running on a network with multiple sinks, the experiment is simulating having multiple organizations attached to a network. This requirement's primary concern is in how information is classified to meet mission objectives when arbitration between

competing objectives is needed. Since this thesis is assuming classification is being done separately from routing, the test is simple. The system passes the multiple sink test by proving an ability to route to multiple sinks.

Since the distributed requirement is fulfilled in proper design and implementation, it is not tested experimentally. The remaining four requirements must be seen during experimentation in Chapter IV. The first test is satisfied when the decision to drop traffic is no longer being made arbitrarily. The second test is satisfied when the network resource, bandwidth, remains close to being optimally utilized. The third test is passed when the system demonstrates the ability to adapt to the network in times of bandwidth flux. Finally, the fourth test shows that the system will extend to multiple sink nodes and therefore multiple organizations.

Summary

This chapter outlines a resource allocation mechanism that can assign bandwidth among all the nodes of a network assuming that classification and scheduling messages can assign an information utility value. This task requires an agent-based framework to coordinate among nodes in the network and adjust threshold and reservation policy. To do this, the agent based framework employs a three layer robot control architecture using a UBF at the controller layer. The applicable behaviors, arbiters, and actions coalesce to provide maximum utilization and minimize information overload. Four tests will show that the agents work to reduce arbitrarily dropped packets, improve utilization, adapt to bandwidth flux, and perform in a distributed manner for several organizations. The agent

based resource allocation mechanism works with QoS routing to help maximize information utility in a network.

IV. Analysis and Results

The network information maximization problem requires a range of solutions, but the design of an agent based framework is a vital component. The agent based framework allocates resources according to the information utility associated with an information flow. Through an NS2 implementation, HANC works with a reservation based routing protocol to send information from source to sink. As packets relay through the network, nodes limit resource contention to the highest utility information. After implementing the multi agent system, experiments show improved average utility and bandwidth utilization.

Implementation

HANC must be evaluated through experimentation that proves its ability to pass the four tests in Chapter III. Thus, proper implementation is necessary to match the design requirements stated earlier. Using the NS2 simulator, a network is designed to provide support to the agent based framework. Finally, HANC is designed in C++ to contain the sequencer, coordinator and UBF module as depicted in Figure 3-3.

Simulation Environment

In order to evaluate a network information maximization solution, a suitable test environment is needed. This work's concern with communication networks demands a network simulator. NS2 is a network simulation tool designed to show how packets flow through networks. It connects nodes with links and uses a wide range of communication protocols to simulate communication networks. The flexibility of NS2 and its large

library of protocols is a great advantage in its use. In addition, NS2 is packaged with the Network AniMator (NAM). NAM provides the ability to watch the network simulation for emergent behavior. For testing and demonstration, this becomes very useful. This paper illustrates emergent behavior with screenshots of the NAM.

Network Setup

The networks described in experiments below use very similar topologies. They are hierarchical in design and have unidirectional flows from source to sink. The most basic topology for the network information maximization problem is shown in Figure 4-1. In this basic network, there are interconnected nodes that provide the functions of information sources, decision points, and information sinks.

Packets are sent from information source to information sink using the available bandwidth at links along the way. Decision points connect links and choose what packets pass over the next links. All packets in the network are associated with an information flow and are generated at periodic intervals at the source nodes. The packets are also uniform in size. Their periodicity and size characteristics reflect packet generation in streaming information. At a big picture level, these sources could be sensor input, streaming video from a UAV, or very large file transfers. The packet headers carry information like flow identifiers, utility values and packet size.

While NS2 simulates many protocols, only a few are used in these experiments. At the transport layer, packets are sent in a connectionless fashion using User Datagram Protocol (UDP). UDP reduces the complexity of connection setup and increases the speed of data transmission. It is also a more realistic protocol for multimedia streams,

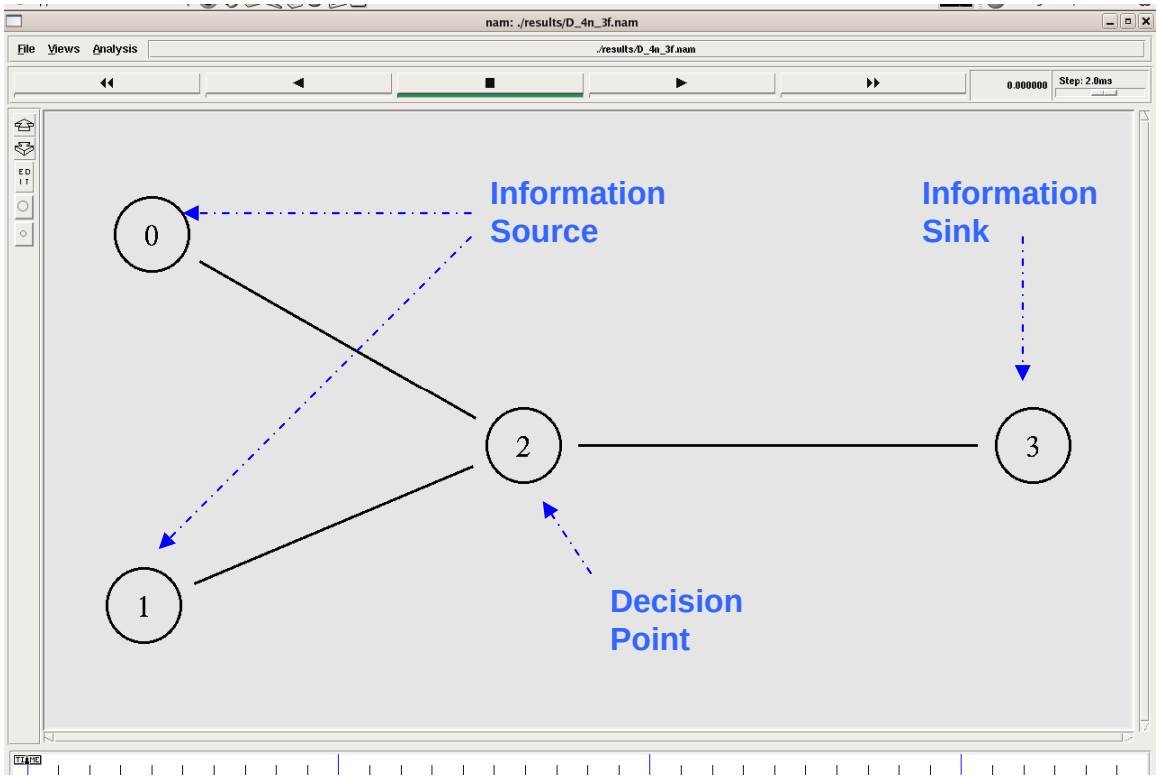

Figure 4-1. Basic Topology for Network Information Maximization in NS2. Nodes 0 and 1 generate information that is relayed through the decision point at node 2. Node 3 collects information from the network.

which the information flows closely emulate. The routing layer uses Internet Protocol (IP) with a reservation based routing scheme.

During network setup, the information flows are assigned to paths through the network using PPRN [8]. PPRN is a multicommodity network flow solver. PPRN generates routing paths that are used to set up reservations in the network, using an information flow's service requirements. During the simulation, information flows produce packetized data that carries the identifier of a particular information flow. The packets traverse links to a router and may enter a classifier upon arrival. The classifier identifies the packet's information flow ID and assigns it to its next hop according to the

preset reservation. If a packet is not associated with an information flow, it is assigned to best effort traffic. A packet's reservation can be rescinded at any node which identifies it as a poor use of resources. Once the packet is classified, it is put in an outbound queue before being sent over a link to its next hop. The queue is designed to drop more low utility packets than higher ones, creating a fast quasi priority queue.

Simulation Management

Some mechanism must manage the simulation, so a controller is put in place to cycle through the several agents in the simulation. Using the backbone of Hopkinson's work [17], the simulator uses a controller called AgentHQ to handle this task. Figure 4-2 shows the pseudo code for the main execution of the AgentHQ. AgentHQ creates the network agents and manages them in an Agent list. AgentHQ also acts as the interface between the middleware and routing layers in the nodes of the network. For instance, the routing topology and reservations are initialized in the AgentHQ. After initialization, the AgentHQ enters into an execution loop that calls each Agent in the Agent list to action for every time step. Typically, this time step is set to 0.002 seconds. A typical action at an information source node is to create another packet for an information flow, while decision point nodes manage relay tasks and sink nodes simply receive messages. Although the AgentHQ allows for a central repository of knowledge, proper implementation ensures that a node only accesses information and procedures for its node.

AgentHQ provides a simulation manager, but it does not perform actions. Figure 4-3 shows the UML diagram of the AgentHQ. Agents are created to perform these

```
AGENTHQ->main_execution( ){
    //network setup, create Agent and add to AgentList
    for each Agent in  SIMULATION {
        HYBRIDCOMMAGENT Agent = new HANCAGENT(location);
        AgentList->add(Agent);
    }
    RoutingTopologyControl->solveRouting( );      //initial routing setup
    for each time_step{                           //main execution loop
        for each Agent in AgentList{
            Agent->action( );
}   }   }
```

Figure 4-2: AgentHQ Main Execution.

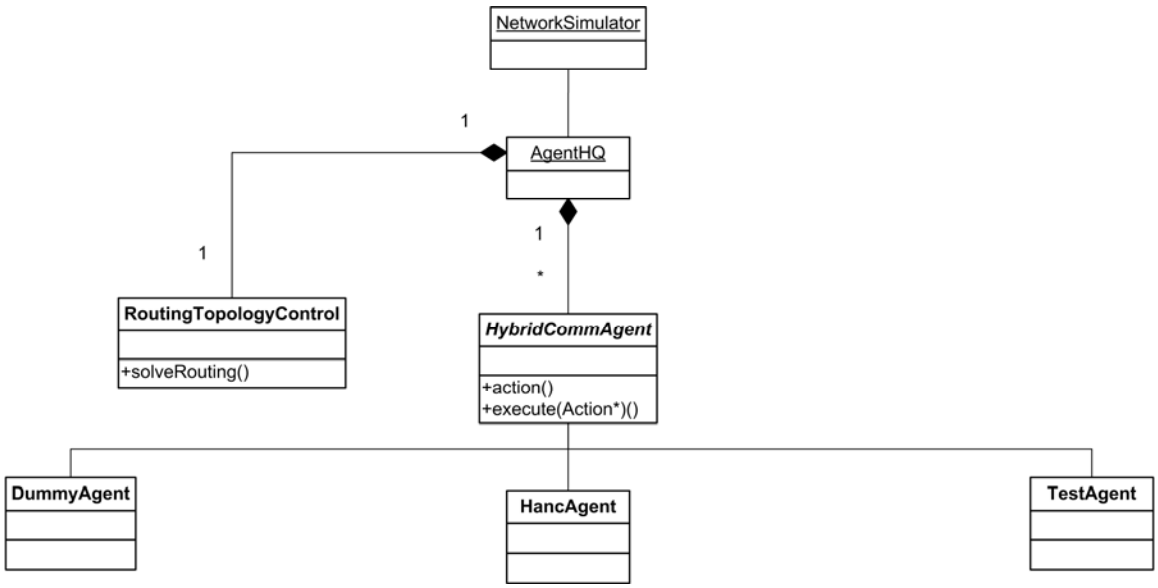

Figure 4-3: UML Diagram for AgentHQ. Agent HQ instantiates the routing functions of the network and builds a network of HybridCommAgents that abstract HANC Agents.

actions, but these agents may be of different types. To allow for heterogeneity, the HybridCommAgent class abstracts the agents while connecting their functionality to the AgentHQ to tie them into the network simulator and the routing layer, also called RoutingTopologyControl. RoutingTopologyControl gives the interface to solve routing when bandwidth flux occurs. It is within this framework that the HancAgent is implemented.

HANC Implementation

Agents are installed at every network node to manage the flow of information. The requirements described in Chapter III pointed to an agent design that could be reactive to the network state. HANC contains a robot control architecture as shown in Figure 3-3. The deliberator layer is an interface to a commander's input through the NTO. The sequencer layer uses network tasking and state analysis to choose a set of behaviors in the controller layer. The controller layer uses these behavior sets to optimize the network for a particular task. The task for network information maximization calls on the HANC's ability to maximize utility and utilization of network resources and requires a specific implementation of the controller layer of the robot control architecture.

HANC implements a rudimentary sequencer, coordinator and UBF Module. A Command Fusion arbiter arbitrates the generated action from five behaviors. The first two are MinThreshold and MaxThreshold, which manage the node's utility threshold. The next two are SendRaise and SendLower, which manage the agent's coordination with its neighbor node though Raise Threshold and Lower Threshold messages. The fifth behavior is HighFlux. HighFlux reacts to the perception of a failed communications node

by triggering an action to repair routing reservations. These five behaviors access a node specific state class to generate their actions.

The main execution of the simulation involves network setup and routing initialization. After nodes, their agents and their routing tables are configured, a central control agent governs the simulation by selecting nodes one at a time in an endless execution loop. When each node is selected, the node is prompted to take an action. The controller layer selects an action when prompted and the agent performs it.

Experiments

To validate the HANC multi-agent system, four tests were prescribed in Chapter III, including the Arbitrary Drop Test, the Utilization Test, the Bandwidth Flux Test, and the Multiple Sink Test. The network simulated experiments to pass these tests.

Experiment 1 – Simple Network

The arbitrary drop test was designed to test an agent's ability to decide what information flows to drop and what to keep. The control experiment runs a network using only priority queues as a means of choosing what to drop. The agent attempts to match or better the results of a priority queue implementation.

A run of the experiment is shown in Figure 4-4. Two information sources send the packets of an information flow to a destination or sink node (4-4a). Default behavior results in the dropping of lower utility packets (4-4b). At 2 seconds, a third information flow vies for network resources. As it has a higher utility, the packets from the third information flow keep QoS guarantees as lower utility flows lose them. Figure 4-4c shows the highly congested network as the network stabilizes.

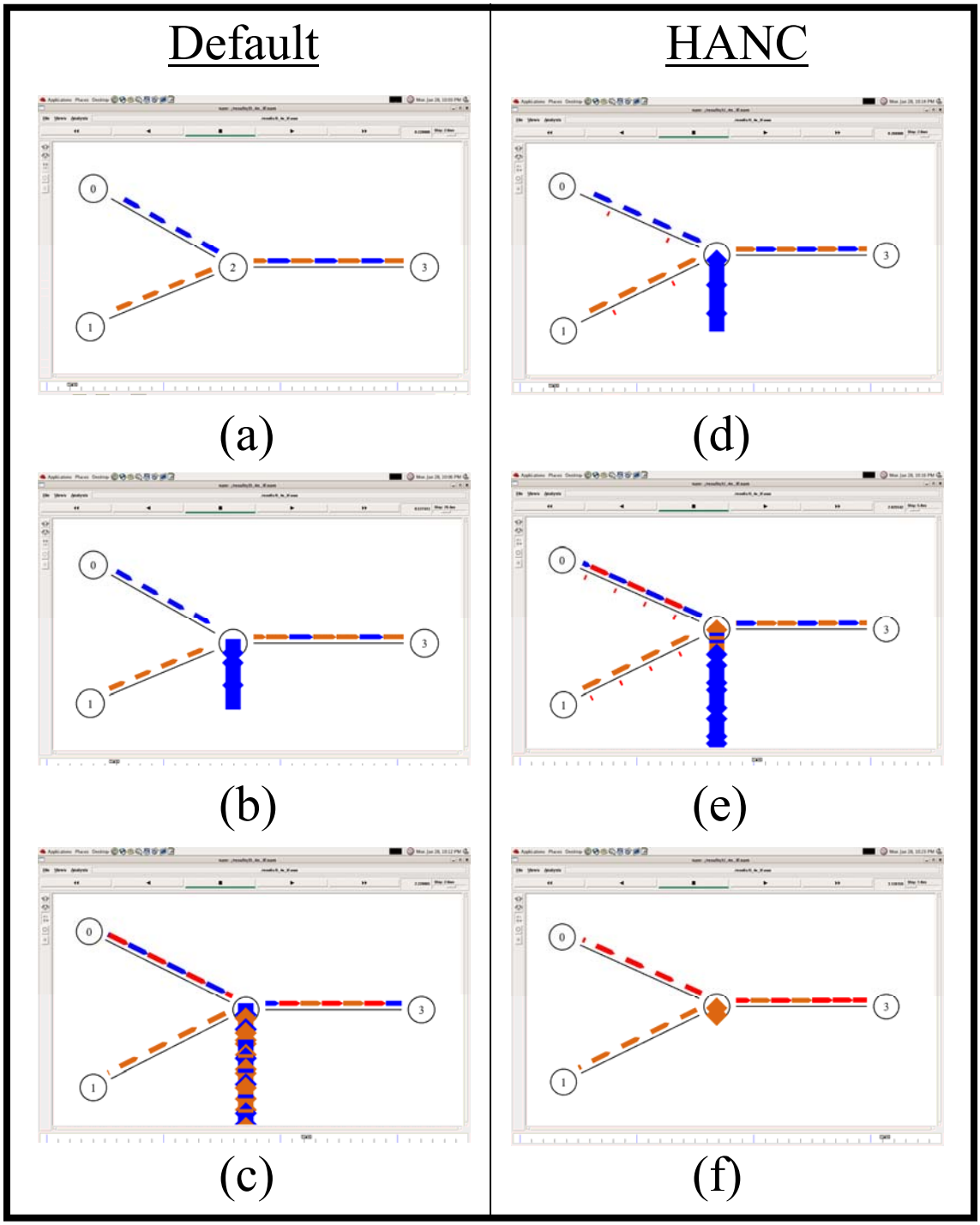

Figure 4-4. Simple Network Experiment, Default and Using HANC. In default mode, nodes forward packets (a) until they are overload and begin to drop less important flows (b) and (c). Using HANC, the network allows moderate packet drops to ensure high link utilization (d), but during overload, agents coordinate (e) to reduce excess flows (f).

In the experiment, HANC maintains the priority queue from the default mode. When information overflows at network relays, the network stabilizes by providing a degraded quality of service to lower utility flows (4-4d). As the third flow enters the network, the network becomes greatly congested, resulting in increased coordination to raise threshold values (4-4e). Eventually, the network adapts to the increased congestion by reducing lower utility flows and allocating bandwidth to the highest utility flows (4-4f). Occasionally, lower utility flows reenter the network to vie for bandwidth.

Figure 4-5 shows how the agents work to achieve a higher average information utility at the sink node than the network of priority queues. The calculation used to determine average information utility divides the total utility of all received packets at each time step divided by the number of packets received at the sink node. In addition, the agent passes the equivalence condition of the utilization test. Table 4-1 shows how utilization of the congested link is identical whether using HANC or default methods.

Experiment 2 – Expanded Network

In the second experiment, the network was expanded to include twice the nodes and an extra information flow. This time, node 4 produced the highest utility flow and node 1's flow produced the second highest utility. Node 0 produced the lowest utility flow, but it is connected directly to the bottleneck to witness how proximity can affect its inclusion into the average information flow. The purpose of the experiment is to ensure that the agents can propagate information requirements through a hierarchy of nodes. Again, the arbitrary drop and utilization tests are evaluated against the network.

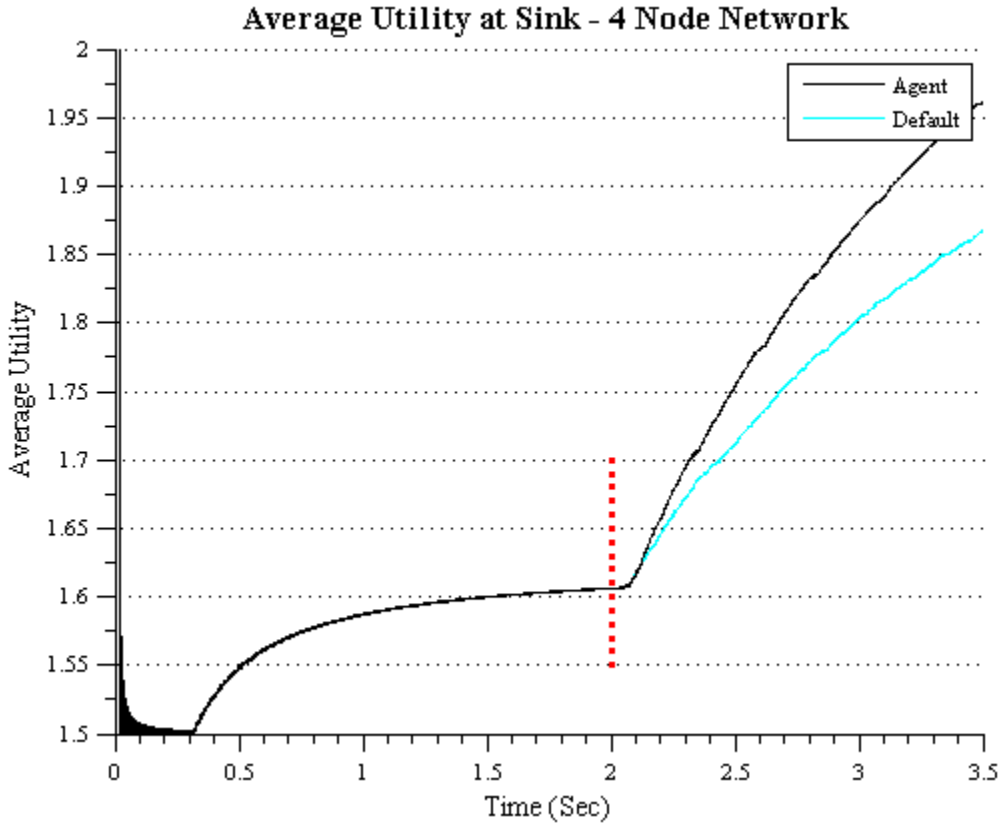

Figure 4-5: Average Utility Comparison in a 4 Node Network. The agent outperforms the default method after the introduction of a mid range utility flow at 2 seconds (dotted line).

Table 4-1: Utilization Comparison over Congested Link in 4 Node Network.

Time	**Utilization (Default)**	**Utilization (HANC)**
0.616	0.121499594	0.121499594
1.226	0.123241232	0.123241232
1.811	0.123809359	0.123809359
2.471	0.124127378	0.124127378
3.026	0.124287426	0.124287426
3.49975	0.124383885	0.124383885

Experiment 3 is demonstrated in Figure 4-6. The network passes all information through a bottleneck link (4-6a). As network links overload, lower priority packets drop and higher utility packets get through. Figure 4-6b shows this default behavior. At 2 seconds, a fourth information flow appeared at node 3. As time went on, the two highest utility flows captured the contested link. Meanwhile, lower utility flows still used network resources in a futile attempt to cross the bottleneck link (4-6c).

Using the agent based framework, an agent at the bottleneck (node 6) coordinated with source nodes to maintain high utilization on the bottleneck link (4-6d). At the introduction of a fourth information flow, the highest information flows dominate the use of the contested link, so the HANC system coordinates to reduce bandwidth utilization on the links used by lower utility flows (4-6e). In Figure 4-6f, the lower utility flows are shut down. The agent based framework succeeds in producing higher average utility at the sink node, while maintaining equivalent utilization at the bottleneck. Table 4-2 shows the utilization result, while Figure 4-7 shows the average utility comparison.

Table 4-2: Utilization Comparison over Congested Link in an 8 Node Network.

Time	**Utilization (Default)**	**Utilization (Agent)**
0.619	0.120910743	0.120910743
1.814	0.123604603	0.123604603
2.464	0.123972707	0.123972707
3.034	0.124165705	0.124165705
3.499	0.124276579	0.124276579

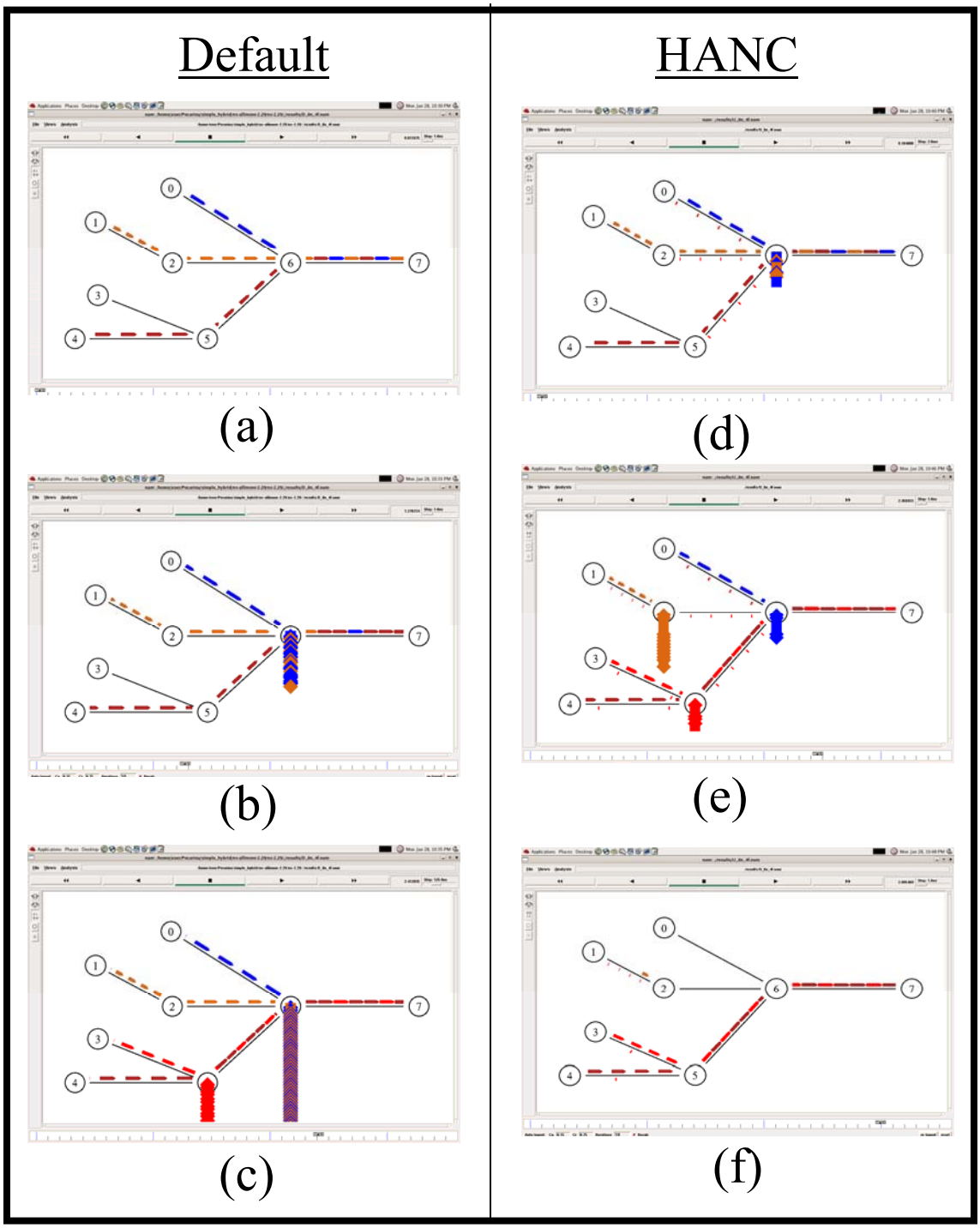

Figure 4-6. Extended Network Experiment, Default and Using HANC. In default mode, nodes forward packets (a) until they are overload and begin to drop less important flows (b) and (c). As in the simple network, the HANC system allows moderate packet drops for high utilization (d), but coordinates during overload (e) to reduce excess flows (f).

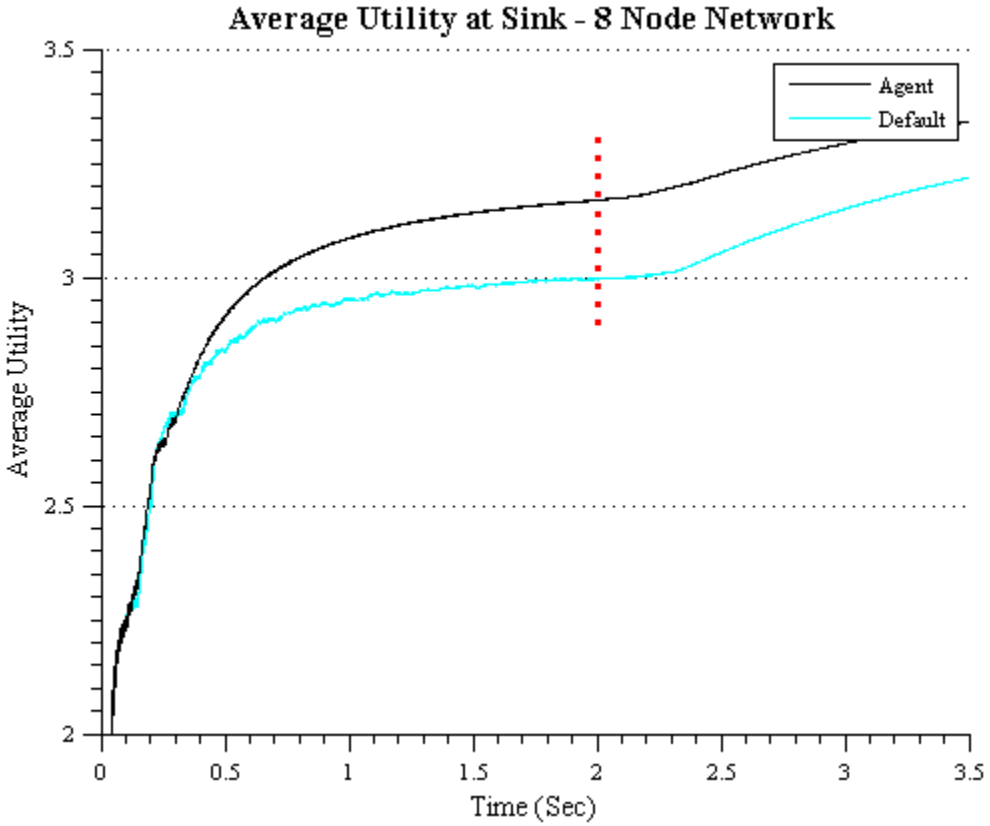

Figure 4-7 Average Utility Comparison in an 8 Node Network. The agent outperforms the default method after the introduction of a mid range utility flow at 2 seconds (dotted line).

Experiment 3 – Multiple Sink Network

A third experiment shows the importance of these results. In this network, shown in Figure 4-8, a second sink node is added. A fifth information flow transmits information from node 2 to node 8 (the alternate sink). Sent as best effort traffic, it can be thought of as a separate organization that uses the same communications infrastructure. As compared with the information flows of the main network, it has no value, although the main network has no jurisdiction to manage it. This experiment is designed to validate the improvement condition of the utilization test and the multiple sink test.

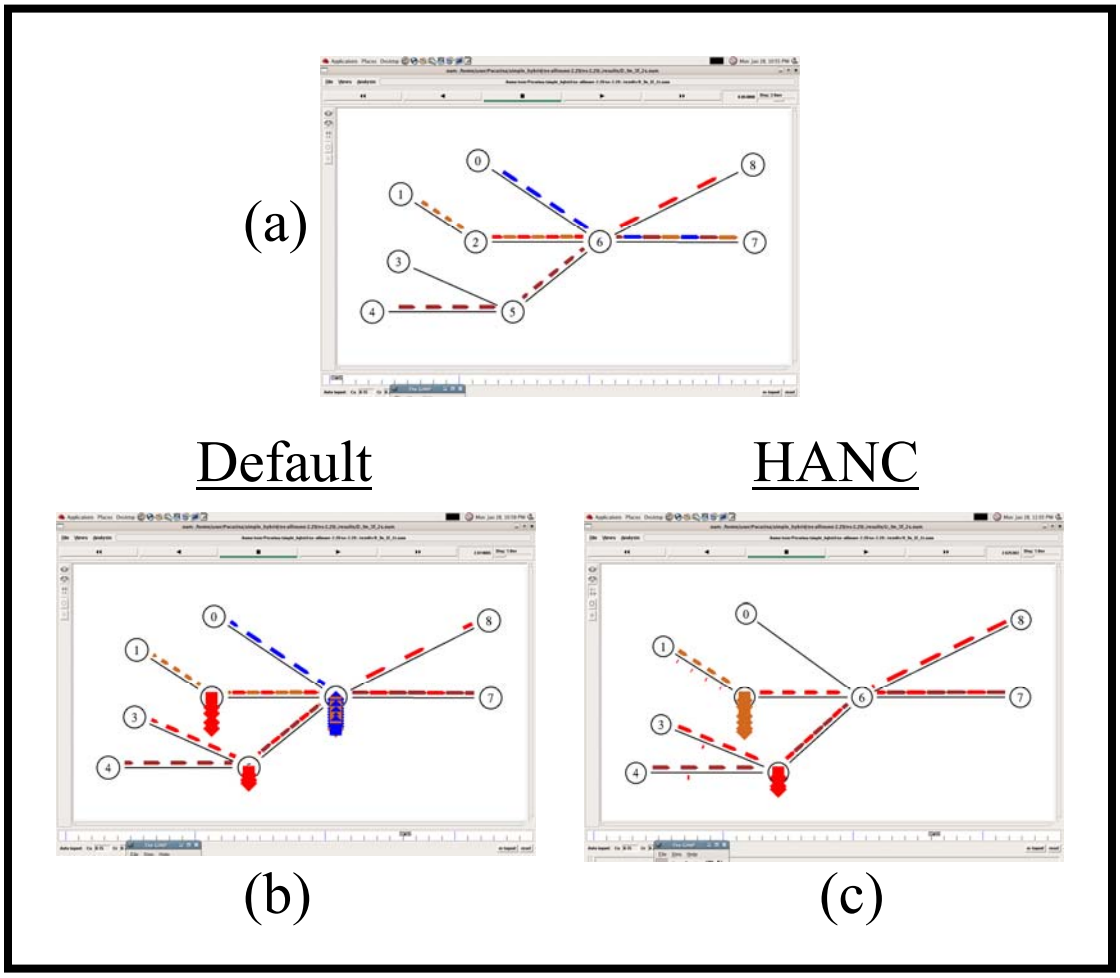

Figure 4-8: Multiple Sink Network Experiment, Default and Using HANC. The network routes an alternate flow to an alternate sink using the resources of the primary network (a). Default operation results in underutilization of the alternate link, while HANC removes unused flows to improve resource usage.

The problem, shown in the default run on this network in Figure 4-8b, is that when network congestion is highest, the alternative flow loses to main flows for bandwidth allocation of the contested link between node 2 and node 6. Congestion here forces poor utilization and packet throughput on the link from node 6 to node 8. In the agent based run, Figure 4-8c, lower utility flows reduce traffic flow when the network is congested, opening the link for alternative flows. In this run, the link from 6-8 is more

heavily used. Figure 4-9 shows that despite this improvement, average utilities at the main sink node are still higher with the agent based framework, passing the arbitrary drop test. The utilization test is also passed, because utilization is the same on the bottleneck link in both the default and agent based runs (Table 4-3). The most powerful result is in Figure 4-10, however, where the utilization of the link from node 6 to 8 is dramatically improved.

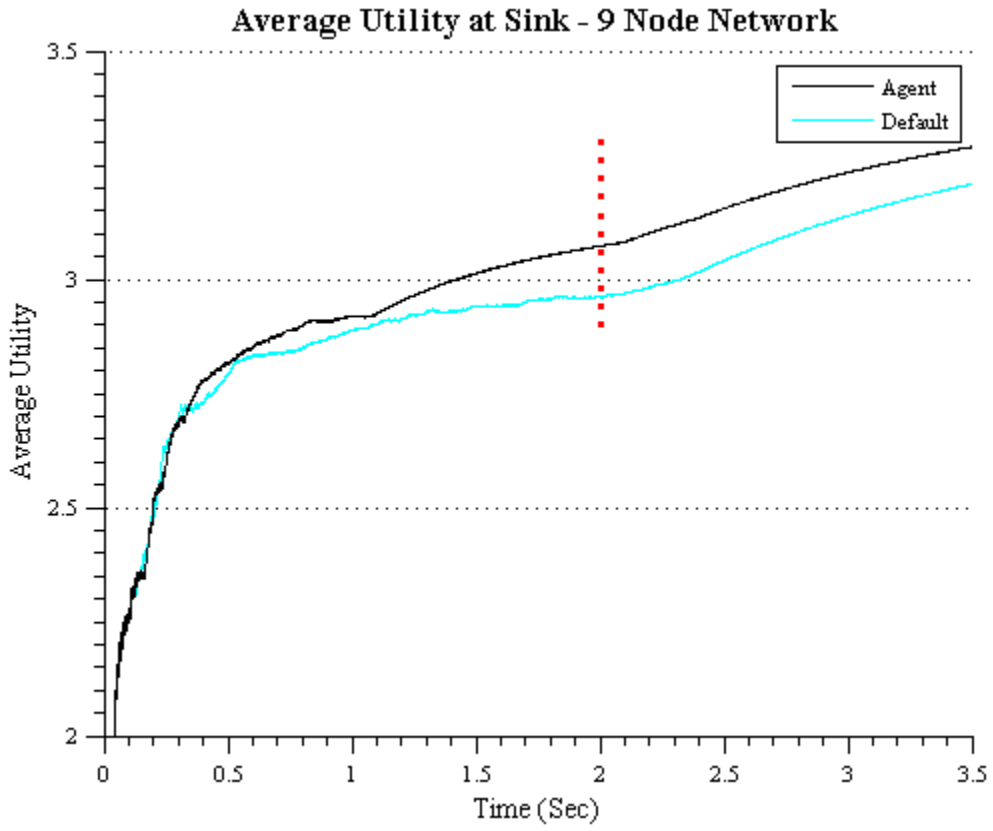

Figure 4-9: Average Utility Comparison in a 9 Node Network. The dotted line at 2 seconds shows the introduction of a higher utility flow to the network.

Table 4-3: Utilization Comparison over Congested Link in a 9 Node Network

Time	Utilization (Default)	Utilization (Agent)
0.619	0.120910743	0.120910743
1.274	0.123013148	0.123013148
1.814	0.123604603	0.123604603
2.464	0.123972707	0.123972707
3.034	0.124165705	0.124165705
3.499	0.124276579	0.124276579

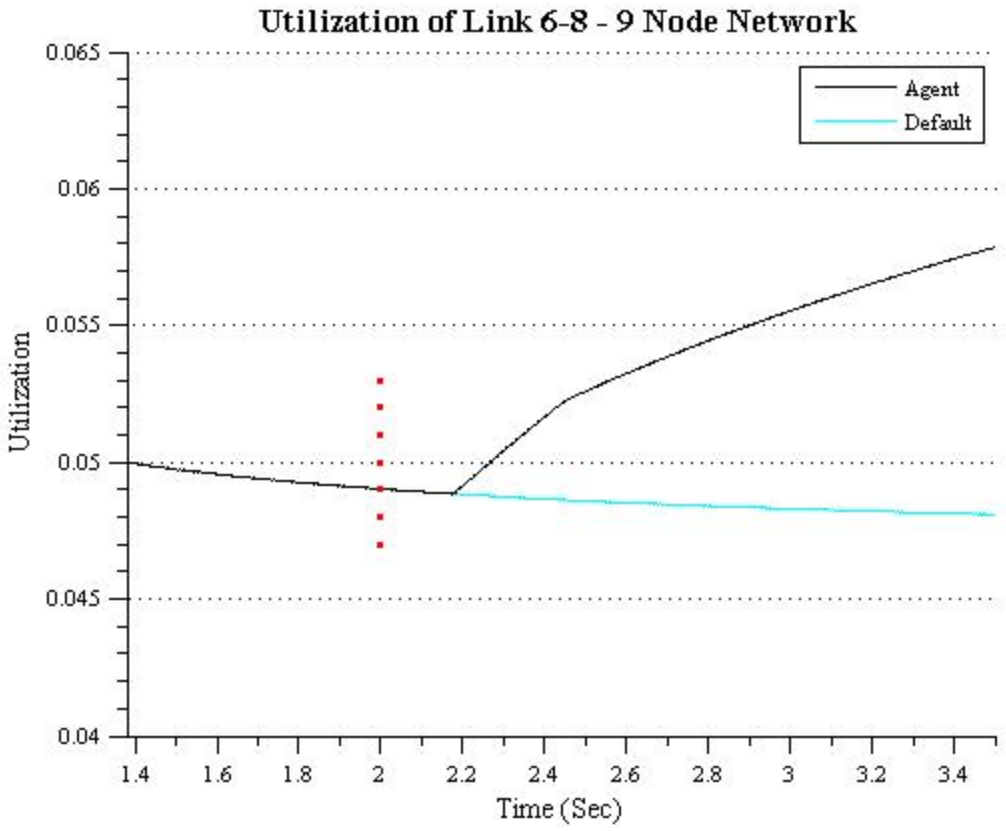

Figure 4-10: Improved Utilization of Link to Alternate Source. The dotted line at 2 seconds shows the introduction of a higher utility flow to the network.

Experiment 4 – Bandwidth Flux Test

The fourth experiment attempts to pass the bandwidth flux test. In this experiment, the network must adapt to the loss of a communications node to maintain QoS for high utility traffic flows. The network in Figure 4-11 is built to test this. Two source nodes, 0 and 1, send information flows to node 8. They route information along two paths. Throughout the simulation, nodes communicate their existence using ping messages. Each node tracks the last time it heard from its neighbor nodes. If the time elapsed since the last node update exceeds a timeout period, the UBF selects an action to solve the routing problem without the failed node.

The path from node 1 initially routes information through node 3, as shown in Figure 4-11a. While the nodes are not sending meaningful information, they send alive pings to notify neighbor nodes of their existence (4-11b). At 0.998 seconds into the simulation, node 3 fails, causing traffic going through that node to be lost (4-11c). Since node 3 fails, it no longer sends pings to neighbor nodes, and after the timeout period, (set arbitrarily in this scenario to 0.01 seconds) the node generates an action to resolve the routing without node 3. The resolved network goes around node 3 (4-11d). At 1.1 seconds, node 3 comes back into the network, sending alive pings yet again (4-11e). After node 3's first ping message is received by a neighbor node, the network routing topology is resolved. Figure 4-11f shows the restored network. This behavior demonstrates HANC's ability to pass the bandwidth flux test.

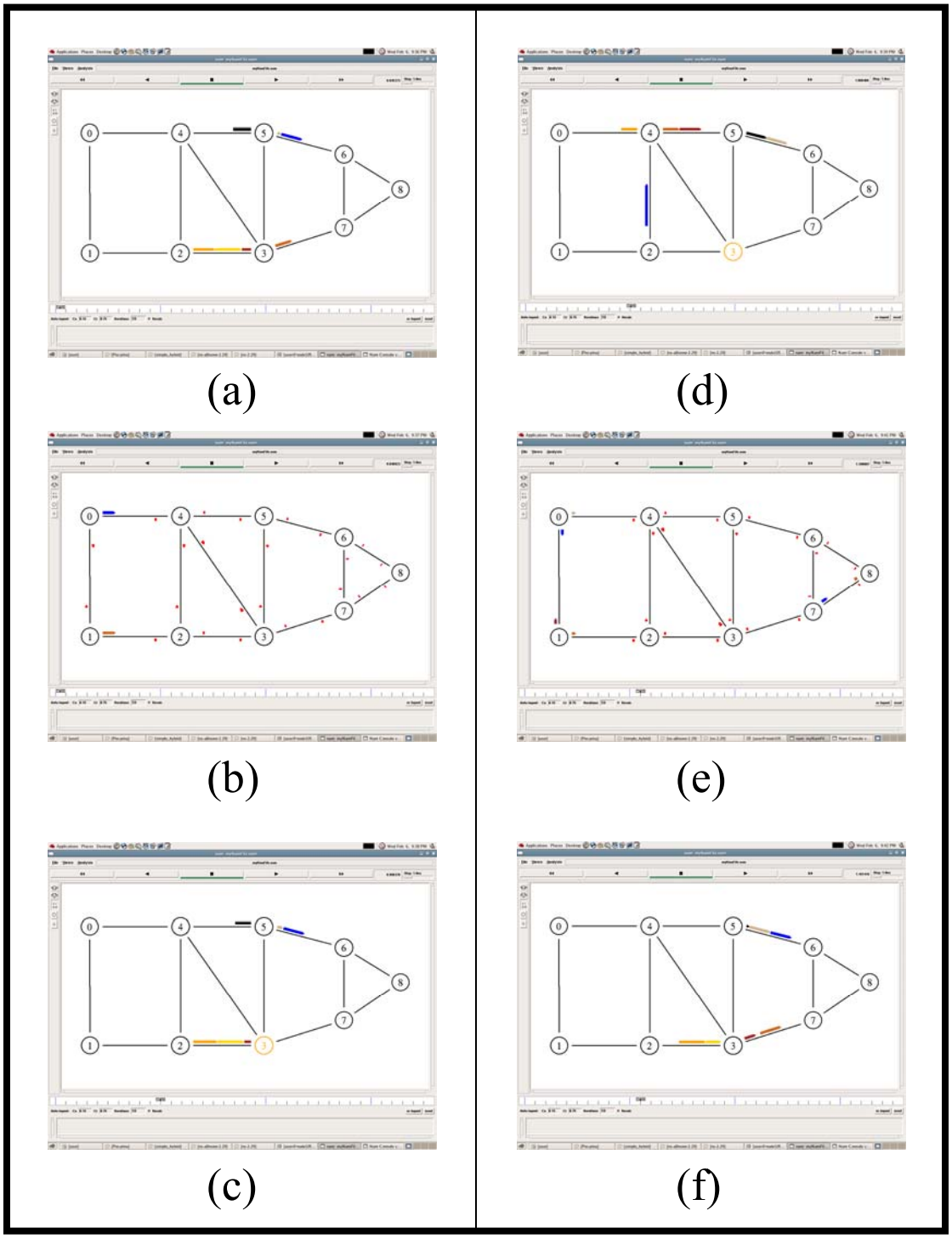

Figure 4-11: Demonstration of Bandwidth Flux Adaptation. Nodes route information (a) and send alive pings (b) during normal operation. If a node fails (c), the HANC agents act to repair the network routing (d). As the failed node returns (e), the agents act to use the newly available links (f).

Analysis

The agents employed in the three experiments improve the ability to use resources efficiently, playing a part in maximizing the average utility of a network. The first two experiments demonstrate the ability of the agent based framework to pass the arbitrary drop test and the utilization test. The third test shows how network resources can be better utilized in a highly congested network, opening poorly used resources for alternative traffic flows. The alternate sink in the third experiment shows that the agent based framework passes the multiple sink test. The fourth experiment demonstrates the capability to react to periods of bandwidth flux.

The experiments in this paper show an improved network in terms of balancing information utility and bandwidth utilization. Likewise, the agent has been designed to handle this optimization task, accessing an appropriate library of behaviors. The experimental scenarios in this section would likely arise in situations where a network sink is overloaded with information and packet losses are high, like when networks experience periods of peak traffic. A sequencer must recognize the signs of impending periods of peak traffic and could build its network agents to perform resource allocation tasks accordingly.

Summary

Commanders desire greater amounts of useful information in terms of mission objectives. While the network information maximization problem requires a range of solutions, the design of an agent based framework is a critical addition. This chapter described the benefits of the agent based framework in resource allocation. Through an

NS2 implementation, the agent works with a reservation based routing protocol to send information from source to sink. As packets relay through the network, nodes limit resource contention to improve average utility and create a network with smarter bandwidth utilization. Yet, as is discussed next, this result is only a small part of the capability of the agent based framework for network control.

V. Conclusions and Recommendations

Research Summary

With increased reliance on communications to conduct military operations, information centric network management becomes vital. OSD's study of information management for net-centric operations list the need for tools for information triage (based on relevance, priority, and quality) to counter information overload, semi-automated mechanisms for assessment of quality and relevance of information, and advances to enhance cognition and information understanding in the context of missions [30]. This study provides impetus to providing a solution to the network information maximization problem.

Maximizing information utility to match mission objectives is a complex problem that requires a comprehensive solution. That solution covers a gamut of research areas. In information classification, the problem involves choosing proper classifiers to determine information value. This paper proposes mission association, timeliness and user input as possible sources. In scheduling, information maximization can be mapped to the restless bandit problem. The solution creates an index for every information flow in the network and selects an approximate solution with a greedy algorithm. In resource allocation, a multi agent system selects actions that manage information flows based on the indexing solution. In QoS support, information flows are given end to end service guarantees through the reservation scheme. Solutions in all of these research areas contribute to maximize average utility.

Of these research areas, the resource allocation mechanism provides a framework to build the entire solution. Through an agent based mindset, the lessons of robot control

architecture are applied to the network domain. The task of managing information flows is achieved in a three tiered mindset. At the top tier, a deliberative planner interfaces with commanders to develop network tasks from mission objectives. The deliberative planner decomposes the mission objectives into network goals. The middle tier, the sequencer, uses these goals and network state to select a library of behaviors that network agents employ to achieve local tasks. In a distributed fashion, nodes coordinate among each other to allocate resources appropriately. At the bottom tier, the controller layer of each agent reacts quickly to the observed state of the network, adjusting policy in dynamic fashion. Using the UBF, agent behaviors combine actions through an arbiter to select a composite action. The action is executed using the available actuators, including message passing, information utility threshold adjustments, and information flow reservation management. The agents employed in the network domain describe a multi agent system that aligns with Cao's and Dudek's taxonomies for multi robot systems. The alignment validates this researches claim that the agent based framework is sound theory.

Research Conclusions

This work presents a distributed agent based framework around the requirements laid out in Chapter I, tailored to the network information maximization problem. The specific purpose for this implementation is to maximize average utility while using resources as efficiently as possible. Chapter IV described the implementation of the agent that could perform this task. Using NS2 to simulate information flows, experimentation showed that the agent based framework passes the arbitrary drop test, outperforming a simple method of priority queues. In addition, it was shown that network resources could be allocated smartly in this framework, as the agents passed the utilization test. Then, the

agent framework performed in a distributed manner with multiple sinks. Finally, the HANC multi agent system proved its ability to react during periods of bandwidth flux.

Future Research Recommendations

This research points to a cornucopia of research areas in the field of network information maximization. Further study of information classification is warranted to confirm the theory that information flows can be classified during network execution without slowing the routing process. This paper proposes mission association, timeliness and user input as attributes, but future research may find other classifiers.

Further work can be done in the scheduling domain to implement the adaptive greedy algorithm for information flows. The object of such study would be to create an index using at least three attributes during network execution. As network execution progresses, indices should change as information flows reach deadlines for timeliness. Along the same line of thinking, as mission goals change or users give feedback, the scheduler should produce different values for information utility.

The most exciting extension to this work requires further development of the agent based framework. With HANC in place and a few behaviors implemented, other network optimizations can be attempted by adjusting behavior sets. The sequencer component of HANC can be expanded to automate parameter selection in the behavior sets. Currently, parameters are set using trial and error techniques, but the addition of a neural network or genetic algorithm at the primitive sequencer layer would empower the agent based framework. For further expansion, an addition of a deliberative planning interface would demonstrate how the NTO process could be implemented in a network of agents. Once an interface is devised, one could watch as network agents perform tasks to

meet the commander's intent in the network. Currently, this research is continuing to attempt to incorporate an encryption mechanism to demonstrate HANC's ability to adapt to changing security requirements. Another application of the agent based framework is fault tolerance and topology control. As communication links increase and decrease, the agent based framework could carry a set of behaviors that could vote how to adapt to the network's flux, including physical movement. An agent could reset its topology if the perception of future quality is bleak, or could choose to remain in its current topology if the cost of moving is too high. If node mobility is possible, a node might even vote to move to another area where links have greater quality. The addition of other network control mechanisms is a significant expansion area.

Final Remarks

This paper broke ground in the network information maximization problem by identifying several key areas of future study. The core implementation of this thesis, the agent based framework, provides the basic architecture for the entire information maximization system. Future research should be able to extend this framework and provide constant improvement in this research area. Yet the strength of this work may not be in its ability to route information to maximize average utility. Although this may be useful in managing networks during periods of peak load, alternative applications abound for the agent based framework. The agent-based framework makes a significant contribution to network management. In a future that includes a cyberspace commander issuing an NTO to match mission objectives, the agent based framework paves the way for the cyberspace commander to operationalize the network.

Appendix A: Pseudo Code for Additional Behaviors

Chapter III described three behavior pairs. The first pair, MinThreshold and MaxThreshold, were described in pseudo code in that chapter. The remaining pairs, SendRaise/SendLower, HighFlux/LowFlux are described in Figures A-1 and A-2 respectively.

```
Action SENDRAISE genAction( ) {            //called by Composite behavior
    ACTION action = new ACTION;
    int droppedPacketTolerance = 2;
    if (State->getDroppedPackets( ) > droppedPacketTolerance) {
        action->setSendRaiseThresholdMsg(true);    //Set the action attributes
        action->setSendLowerThresholdMsg(false);  //Set the action attributes
    }
    action->setVote(State->getDroppedPackets( ) );  //Set a vote
    return action;                                  //Return an action
}

Action SENDLOWER genAction( ) {            //called by Composite behavior
    ACTION action = new ACTION;
    if (State->getUtilizationTrend( )< 0)    //if outbound queue sizes are dropping
        action->setSendLowerThresholdMsg(true);  //Set the action attributes
        action->setSendRaiseThresholdMsg(false);  //Set the action attributes
        action->setVote(10);                      //Set an arbitrary vote
    return action;                                //Return an action
}
```

Figure A-1: SendRaise and SendLower Action Generators. The two behaviors watch queue sizes to determine if coordination must intervene to manage information flows.

```
Action HIGHFLUX genAction( ) {              //called by Composite behavior
    ACTION action = new ACTION;
    double fluxTolerance = .2;                          //long delay
    if (State->getTimeElapsedSinceUpdate( ) > fluxTolerance)
        action->setSolveRoutingTopology(true);          //Set the action attribute
    else if (State->newNode( ) )
        action->setSolveRoutingTopology(true);          //Set the action attribute
    else
        action->setSolveRoutingTopology(false);         //Set the action attribute
    action->setVote(2 );                                //Set a vote
    return action;                                      //Return an action
}

Action LOWFLUX genAction( ) {               //called by Composite behavior
    ACTION action = new ACTION;
    double fluxTolerance = .02;                         //short delay
    if (State->getTimeElapsedSinceUpdate( ) > fluxTolerance)
        action->setSolveRoutingTopology(true);          //Set the action attribute
    else if (State->newNode( ) )
        action->setSolveRoutingTopology(true);          //Set the action attribute
    else
        action->setSolveRoutingTopology(false);         //Set the action attribute
    action->setVote(2 );                                //Set a vote
    return action;                                      //Return an action
}
```

Figure A-2: HighFlux and LowFlux Action Generators. The two behaviors activate the solve routing action when nodes seem to disappear or when new nodes appear in the network.

Appendix B: Software and Procedures for Set Up and Simulation

A. (Optional) Install the latest version of Cygwin in order to provide a Linux-like environment for Windows.

B. Install, configure and compile Network Simulator version 2 to be run inside of Cygwin or Linux.

C. Install Network Animator (NAM) to run on NS2 output files.

D. Copy the HANC Agent Hybrid Communications files into the proper directories of NS-2.

E. Add the new files into the Makefile.

F. Load PPRN executable into the NS2 root directory.

G. Build an NS-2 script using the tcl language. A script generator is available among the HANC files.

H. Run NS2 on the NS2 script. Example command at command line is
home/ns-2.29/> ./ns ./nscript.tcl

I. Run NAM to view the NAM output file. Example command is
home/ns-2.29/> ../nam-1.11/nam MyNamFile

Bibliography

1. Bass, Samuel D. "*A Model for Managing Decision-Making Information in the GIG-Enabled Battlespace*" Air and Space Power Journal 21, no. 2 (Summer 2007): p. 6-8, http://www.airpower.maxwell.af.mil/airchronicles/apj/apj07/sum07/sum07.pdf.

2. Berkhin, P. "*Survey of clustering data mining techniques.*" Technical Report. Accrue Software, San Jose, CA. 2002.

3. Bertsimas, Dimitris and Jose Nino-Mora. *"Restless bandits, linear programming relaxations, and a primal-dual index heuristic."* Operations Research. v. 48 n 1 p. 80-90. 2000.

4. Braden, R., L. Zhang, S. Berson, S. Herzog, and S. Jamin, *"Resource ReSeVation Protocol (RSVP) Version 1, Functional Specification."* RFC 2205, IETF, Sep 1997.

5. Braitenberg, V., 1984, "*Vehicles: Experiments in Synthetic Psychology*", MIT Press, Cambridge Massachusetts.

6. Brooks, R.A., "*A Robust Layered Control System for a Mobile Robot*." IEEE Journal of Robotics and Automation, Vol. 2, No. 1, p. 14-23, Mar 1986.

7. Cao Y. Uny, Alex S. Fukunaga, and Andrew Kahng. *"Cooperative mobile robotics: Antecedents and directions. Autonomous Robots."* 4(1) p. 7-27, 1997.

8. Castro, J. *PPRN 1.0 User's Guide*. Technical Report. Universitat Politknica de Catalunya. Sep 1994.

9. Dudek, Gregory, Michael Jenkin, Evangelos Milios, and D. Wilkes. *"A Taxonomy for Swarm Robots"*. Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems, p. 441–447. Yokohama, Japan, 1993.

10. Fall, K. and K. Varadhan. *The ns Manual*. Technical Report. UC Berkeley, LBL, USC/ISI, and Xerox PARC. Feb 2008. http://www.isi.edu/nsnam/ns/doc/ns_doc.pdf

11. Gat, Erann. "*On Three-Layer Architectures*." Artificial Intelligence and Mobile Robots. Pasadena, CA. AAAI Press, (1998).

12. Harmon, D., "*Overcoming TCP Degradation in the Presence of Multiple Intermittent Link Failures Utilizing Intermediate Buffering*", Technical Report, Air Force Institute of Technology: Wright-Patterson AFB, 2007.

13. Hintz, K. and J. Malachowski, "*Dynamic goal instantiation in goal lattices for sensor management*," Signal Processing, Sensor Fusion, and Target Recognition XIV; Ivan Kadar; Ed., Proc. SPIE Vol. 5809, p. 93-99, Orlando, FL, April 2005.

14. Hintz, K. and G. McIntyre., *"Goal Lattices for Sensor Management."* Proceedings Signal Processing, Sensor Fusion, and Target Recognition VIII, Ivan Kadar; Ed., Proc. SPIE vol. 3720, p. 249-255, Orlando, FL, Apr 1999.

15. Hintz, K.J. and E.S. McVey. "*Multi-Process Constrained Estimation*." In IEEE Transactions on Systems, Man, and Cybernetics, Vol. 21, No. 1 Jan/Feb 1991.

16. Hooper, D., "*A Hybrid Multi-Robot Control Architecture*", Technical Report, Air Force Institute of Technology: Wright-Patterson AFB, 2007.

17. Hopkinson, K., et al., "*EPOCHS: A Platform for Agent-Based Electric Power and Communication Simulation Built From Commercial Off-the-Shelf Components."* IEEE Transactions, 21: p. 11. 2006.

18. Huberman, Bernardo A. and Wu, Fang. *"The Economics of Attention: Maximizing User Value in Information-Rich Environments."* HP Labs, Palo Alto, CA 94304. 25 May 2007.

19. Huang, J. Wang, Y. and Cao, F. *"On Developing Distributed Middleware Services for QoS- and Criticality-Based Resource Negotiation and Adaptation."* The International Journal of Time-Critical Computing Systems. Kluwer Academic Publishers, p. 187-221. Boston. 1999.

20. Joint Chiefs of Staff, *Joint Net-Centric Operations Campaign Plan: Joint Net-Centric Operations Fact Sheet* (Washington, DC: Joint Staff; Command, Control, Communications, and Computer Systems Directorate [J-6]; October 2006), http://www.jcs.mil/j6/c4campaignplan/JNO_fact_sheet.pdf (accessed 8 October 2007).

21. Keshav, Srinivasan. *An Engineering Approach to Computer Networking: ATM Networks, the Internet and the Telephone Network.* p. 446-449. Addison-Wesley Publishing. Reading, Massachusetts.1997.

22. Kwiatkowski, Marek George, Peter. "*A Network Control and Management Framework Supporting Military Quality of Service."* In Military Communications Conference Proceedings, MILCOM 1999. IEEE. P. 1161-1165 vol.2. 1999.

23. Llewellyn, L., "*Distributed Fault Tolerant Quality of Service Routing in Hybrid Directional Wireless Networks*.", Technical Report, Air Force Institute of Technology: Wright-Patterson AFB, 2007.

24. Mathy, L. and D. Hutchison and S. Schmid and G. Coulson. "*Improving RSVP for Better Support of Internet Multimedia Communications*." from ICMCS '99, Florence, Italy, Jun 1999.

25. McIntyre, G. A. *"A comprehensive approach to sensor management and scheduling."* Fairfax, VA: George Mason University, 1998.

26. Mizzaro, S. *"Relevance: The whole history*." Journal of the American Society for Information Science, 48(9), p. 810–832. 1997.

27. Morrison, Clayton T. and Cohen Paul R. *"Noisy Information Value in Utility Based Decision Making.*" UBDM '05 ACM. August 21, 2005. Chicago, Illinois, USA. 2005.

28. "*Multi-Armed Bandit Problem*." http://en.wikipedia.org/wiki/Multi-armed_bandit. accessed 30 Nov 2007

29. Nino-Mora, J. "*Restless bandits, partial conservation laws and indexability*." Adv. Appl. Probab. 33, p. 76–98. 2001.

30. Office of the Secretary of Defense, "*2006 Summer Study: Information Management for Net-Centric Operations."* Vol I. http://www.acq.osd.mil/dsb/reports/2007-04-IM_Vol_I.pdf

31. Papadimitriou, C.H. and J.N. Tsitsiklis. "*The complexity of optimal queueing network control*." Math. Oper. Res. 24, p. 293–305 (1999).

32. "*Quality of Service*." http://en.wikipedia.org/wiki/Quality_of_service accessed 15 Jan 2008.

33. Rosen, E., A. Viswanathan, R.Callon "*Multiprotocol Label Switching Architecture (RFC 3031)*" Jan 2001. http://www.ietf.org/rfc/rfc3031.txt.

34. Secretary of the Air Force. "*SECAF/CSAF Letter to Airmen: Mission Statement*," Air Force Link, 7 Dec 2005. http://www.af.mil/library/viewpoints/jvp.asp?id=192 accessed 8 Oct 2007.

35. Sycara, Katia P. "*Multiagent Systems*." AI Magazine. v. 19 n. 2 , p 79-92. 1998.

36. Wang, J. and G. Beni. "*Distributed computing problems in cellular robotic systems*." In IEEE/RSJ IROS, p 819–826, 1990.

37. Woolley, B., and G.L. Peterson, "*Unified Behavior Framework for Reactive Robot Control.*", ACM Transaction on Autonomous and Adaptive Systems. (submitted).

38. Wynne, Michael W. "*Flying and Fighting in Cyberspace*," Air and Space Power Journal 21, no. 1, p 6-8, Spring 2007. http://www.airpower.maxwell.af.mil/airchronicles/apj/apj07/spr07/spr07.pdf

39. Zhang, L., S. Deering, D. Estrin, S. Shenker, and D. Zappala. "*RSVP: A New Resource ReSerVation Protocol."* IEEE Network, p 8-18, Sep1993.

Vita

Captain John Matthew Pecarina graduated from Central High School in San Angelo, Texas. He received his Bachelor of Science in Computer Science from Angelo State University in December 2001. Immediately after graduation, he was commissioned in the Air Force as a communications officer. In January 2002, Captain Pecarina assumed the duties of a crew commander in the Air Force Space Command Network Operations Security Center at Peterson Air Force Base (AFB). In October 2004, he was assigned to the 608th Air Communications Squadron at Barksdale AFB, supporting the Combined Air and Space Operations Center for USSTRATCOM's Global Strike mission.

Before attending the Air Force Institute of Technology in August 2006, Captain Pecarina served as Aide-de-Camp to the Commander of Eighth Air Force and Joint Functional Component Commander for Space and Global Strike. His next assignment is with the Communications Directorate of Air Mobility Command at Scott AFB, Illinois.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From – To)*
27-03-2008	**Master's Thesis**	Sep 2006 – Mar 2008

4. TITLE AND SUBTITLE

Creating an Agent Based Framework
to Maximize Information Utility

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

John M. Pecarina, Captain, USAF

5d. PROJECT NUMBER
JON # 08-175

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GCS/ENG/08-19

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Office of Scientific Research/NM
Dr. David Luginbuhl
875 N. Randolph, Ste. 325, Rm. 3112
Arlington, Virginia, 22203
(703) 696-6207 david.luginbuhl@afosr.af.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

With increased reliance on communications to conduct military operations, information centric network management becomes vital. A Defense department study of information management for net-centric operations lists the need for tools for information triage (based on relevance, priority, and quality) to counter information overload, semi-automated mechanisms for assessment of quality and relevance of information, and advances to enhance cognition and information understanding in the context of missions. Maximizing information utility to match mission objectives is a complex problem that requires a comprehensive solution in information classification, in scheduling, in resource allocation, and in QoS support. Of these research areas, the resource allocation mechanism provides a framework to build the entire solution. Through an agent based mindset, the lessons of robot control architecture are applied to the network domain. The task of managing information flows is achieved with a hybrid reactive architecture. By demonstration, the reactive agent responds to the observed state of the network through the Unified Behavior Framework (UBF). As information flows relay through the network, agents in the network nodes limit resource contention to improve average utility and create a network with smarter bandwidth utilization. While this is an important result for information maximization, the agent based framework may have broader applications for managing communication networks.

15. SUBJECT TERMS
Network Management, Robot Control Architecture, Information Maximization, Multi Agent systems

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
REPORT	**ABSTRACT**	**c. THIS PAGE**			Dr Kenneth M. Hopkinson (ENG)
U	**U**	**U**	**UU**	122	**19b. TELEPHONE NUMBER** *(Include area code)* (937) 785-3636, ext 4579; e-mail: Kenneth.Hopkinson@afit.af.mil

Standard Form 298 (Rev: 8-98)
Prescribed by ANSI Std. Z39-18